Introduction

Open32 provides functions that let programmers maintain application source code that is common between the OS/2 and Windows 32-bit programming platforms. Open32 functions are entry points to OS/2 functions that correspond to Windows 32-bit functions. The Windows and Open32 Differences section documents the Windows 32-bit and Open32 behavioral differences. Where appropriate, see the Windows 32-bit programming reference of your choice for the specific syntax and usage of these programming interfaces.

Developing Common Source Code

IBM analyzed more than nine million lines of Windows 32-bit code and implemented the most commonly used Windows 32-bit functions and messages in Open32: approximately 750 Windows 32-bit functions and almost all of the Windows 32-bit messages.

The Windows 32-bit programs that you will convert to OS/2 Warp programs fall into one of the two following categories:

Open32 applications are small sample programs that you can convert to OS/2 Warp programs by using only the Open32 functions. IBM evaluated and converted many such programs without any problems. These applications are called Open32 applications because they use only Open32. You can compile the unchanged source with the appropriate header files.

Mixed-mode Windows 32-bit applications are programs that contain both Open32 and other non-OS/2 functions and require separation of common code and platform-specific code into individual source files. Due to the complex nature of applications and because Open32 is a subset of the Windows 32-bit functions, most of Windows 32-bit applications fall into this category.

Summary

To use Open32 for Open32 applications or mixed-mode applications, follow these steps:

1. For mixed-mode applications, see Migrating Mixed-Mode Applications.
2. Get set up to use the Open32 by installing the prerequisites. See Getting Started.
3. If you wish to analyze and migrate existing Windows 32-bit code, use SMART to analyze Windows code and resource files. Then use SMART to migrate the code and resource files to OS/2 code and resource files. Use either SMART or Hyperwise Lite to migrate help files to OS/2 help files. See the following:
   - Analyzing Windows Code
   - Analyzing Windows Resource Files
   - Migrating Windows Code
   - Migrating Windows Resource Files
   - Migrating Windows Help Files
4. To migrate existing Windows icons and bitmaps, use SMART. The *Open32 Programming Guide and Reference* does not cover this topic; see the SMART documentation.
5. If you want to write new code or ensure that your existing migrated code runs correctly, note the:
   - Differences between Windows NT and OS/2: See Windows and Open32 Differences
   - Changes to the OS/2 Resource Compiler: See *OS/2 Warp Tools Reference*
   - Open32 functions: See Open32 Functions
6. Compile and test the code: See Testing.

Migrating Mixed-Mode Applications

This section describes how to migrate mixed-mode applications using Open32 and SMART. SMART provides a porting mechanism, and

Open32 makes the porting process easier. It is essential to have a reasonable amount of understanding of both the Windows 32-bit and OS/2 Warp operating systems to port applications from one platform to another or create applications that can run on multiple platforms. By using Open32 and SMART, you might not need to learn about certain OS/2 topics, such as the Presentation Manager (PM), but you still need to be familiar with the unique characteristics of each operating system.

To port a mixed-mode Windows 32-bit application to OS/2 Warp, you must first analyze the existing Windows 32-bit application so that you can separate the common code from the platform-specific code within the Windows 32-bit application. Use SMART and the Open32 dictionary to analyze the Windows 32-bit application. The analysis identifies areas of code that can and cannot be converted using Open32. At this point, examine the code for possible ways of converting the platform-specific functions to Open32. Many times there are multiple Windows 32-bit functions implementing a single function. Open32 implements only the most commonly used Windows 32-bit functions. Often, the percentage of common source can be increased by modifying the original Windows 32-bit source to use the Open32 functions. However, there are some functions that Open32 does not support (for example, OLE) that must remain platform specific.

Use the SMART analysis to help you separate the Windows 32-bit common and platform-specific code.

When the code is separated, add a header change to the common code section so that you can successfully compile this code. This common code can call platform-specific functions as needed except for those functions mentioned in Common Code Support Caveats.

Common Code Support Caveats

Interoperability does come with certain caveats:

- Do not mix and match operations such as Open32 and OS/2 handles. In other words, do not pass handles that are obtained from Open32 to OS/2 functions and vice versa.

- Try not to mix Open32 and OS/2 functions. For example, do not mix Open32 and OS/2 graphics (GPI) calls. An exception to this rule is that you can create an OS/2 child window from a Open32 window or vice versa.

- Open32 does not provide complete Windows 32-bit function compatibility due to underlying differences in the operating system platforms. For example, the maximum coordinates allowed in Windows 32-bit are not the same as those allowed in OS/2. Open32 supports the maximums imposed by OS/2.

- For Registry functions, note that data storage and retrieval from the Registry requires minor modifications to your code so that it can work on multiple platforms.

- Resource functions are not compatible with Windows. Except for the LoadResource function, which returns a pointer to an OS/2 resource structure, each resource function takes a pointer to a resource structure as a parameter. These functions take and return pointers to OS/2 resource structures, not Windows resource structures. Other functions that fall into this Resource function category include:

    - CreateBitmapIndirect
    - CreateIconIndirect
    - CreateDialogIndirect
    - CreateDialogIndirectParam
    - DialogBoxIndirect
    - DialogBoxIndirectParam
    - LoadMenuIndirect
    - LoadResource

After these changes have been made and the code has been separated, you are ready to use SMART to convert the platform-specific code. This process is described in the SMART documentation and in this guide.

Migrating Mixed-Mode Resource and Help Files

The Open32 Resource and Help functions recognize only OS/2 Resource files and OS/2 Help files. To be able to use your Windows Resource (RC extension) and Help files you need to convert them to OS/2 format using any of the existing converter tools. After conversion to OS/2, you should check the converted Resource and Help files for functionality and look and feel.

SMART converts both Windows Resource and Help files to OS/2 format. While SMART does a good conversion job, you will need to use a viewer to look at the converted files. You can also use the Borland RC converter (available with their OS/2 C++ compiler) or Universal Resource Editor (URE) that is available on the OS/2 Developer's Connection. Hyperwise Lite is another tool that helps you convert Windows help files to OS/2 format. It also includes a viewer. Hyperwise Lite provides many other Rich Text Format (RTF) to OS/2 Help file conversion functions that are not included in Hyperwise today. Because OS/2 Resource and Help functions are different from their Windows counterparts, you might need to further modify your Help and Resource files to obtain similar functionality on both platforms. If your source code is dependent on the structure of these Resource files, you might need to modify your source code for OS/2. Identify this modified code as platform-specific code, because this modification is always necessary.

Also, the converted Resource files need to use OS/2 header files, not Open32 header files. Because Open32 common code expects Open32 header files and Resource files expect OS/2 header files, an include that is shared by the Open32 code and Resource files must avoid using either Open32 or OS/2 header files. In summary, pay close attention to header files that are used by both Open32 common source code and Resource files.

The Windows 32-bit Help and OS/2 Help functionality are dramatically different. Due to the difference in Help behavior, you must understand the limitations of the conversion tools. Information on these limitations is provided in the documentation for the Help conversion tools. You need to understand what needs to be modified in the converted Help files to obtain similar functionality. The macro calls made in RTF files need to be closely monitored because the conversion tools might not support all the RTF macros. Help can be one of the areas that requires fine tuning to get similar functionality across Windows 32-bit and OS/2 environments.

Getting Started

Install the following to use Open32 to analyze and migrate Windows code to OS/2 code:

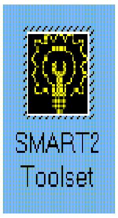
- The OS/2 Warp Version 3.0 Toolkit, which includes Open32 headers and libraries. See the README that accompanies the Toolkit.
- SMART Version 2.1B. See the README that accompanies SMART.
- Hyperwise Lite. See the README that accompanies Hyperwise Lite.
- IBM VisualAge C++ for OS/2 Version 3.0: Call IBM Direct at 1-800-342-6672 to order. Follow the instructions that accompany the compiler to install it.

When you have installed everything, you must shut down and restart your system. Then, for analyzing and migrating Windows 32-bit code, you need to create the Open32 UDMD, see Creating the Open32 UDMD.

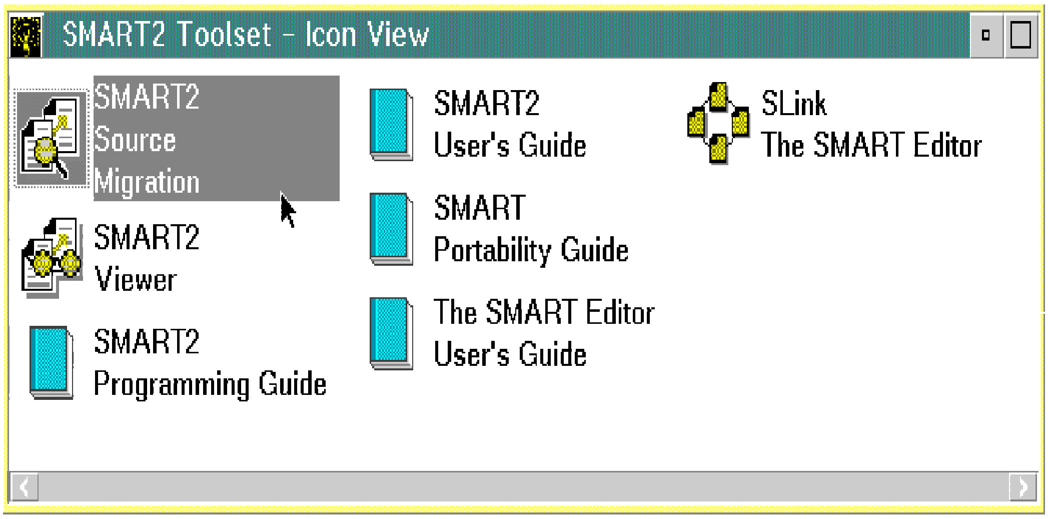
Creating the Open32 UDMD

To analyze and migrate Windows 32-bit code, you need to create the Open32 UDMD as follows:

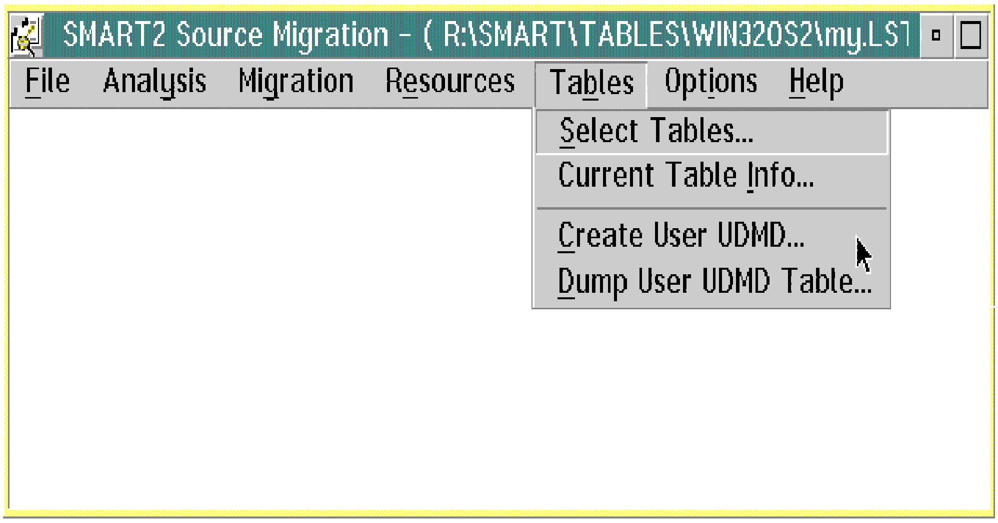
1. Double-click on the **SMART2 Toolset** object to start SMART.



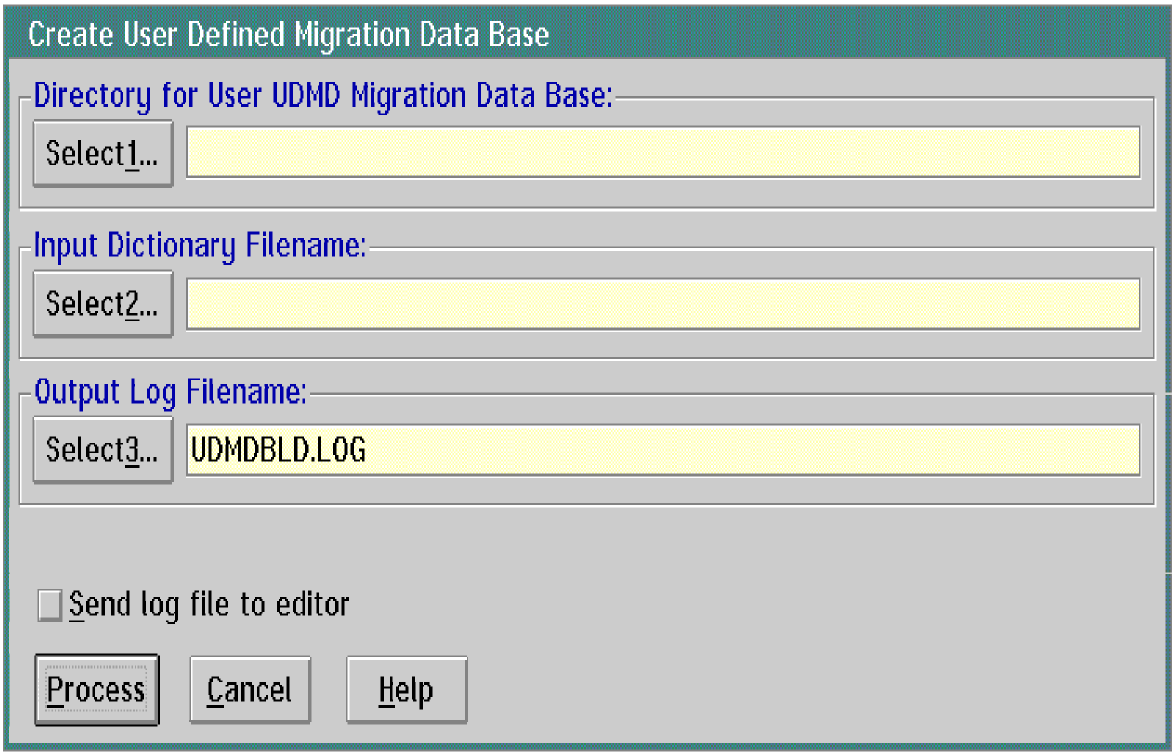
2. SMART displays the SMART2 Toolset - Icon View window. Double-click on the **SMART2 Source Migration** object.



3. SMART displays the SMART2 Source Migration window. Select the Tables pull-down. Select Create User UDMD from the pull-down.



4. SMART displays the Create User Defined Migration Data Base dialog. Click on the **Select1** push button to choose the directory name in which to store the Open32 UDMD.



Create User Defined Migration Data Base

Directory for User UDMD Migration Data Base:

Input Dictionary Filename:

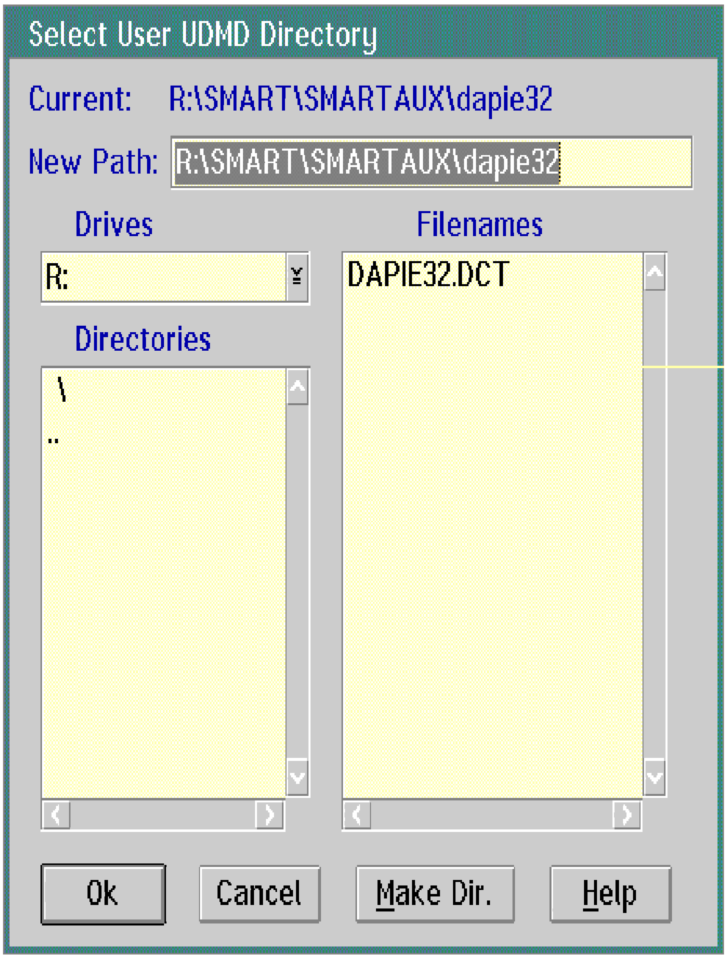
Output Log Filename: UDMDBLD.LOG

□ Send log file to editor

5. SMART displays the Select User UDMD Directory dialog. In the New Path: entry field, type the subdirectory:

```
SMART\SMARTAUX\DAPIE32
```

Click on the **Ok** push button.



Select User UDMD Directory

Current: R:\SMART\SMARTAUX\dapie32

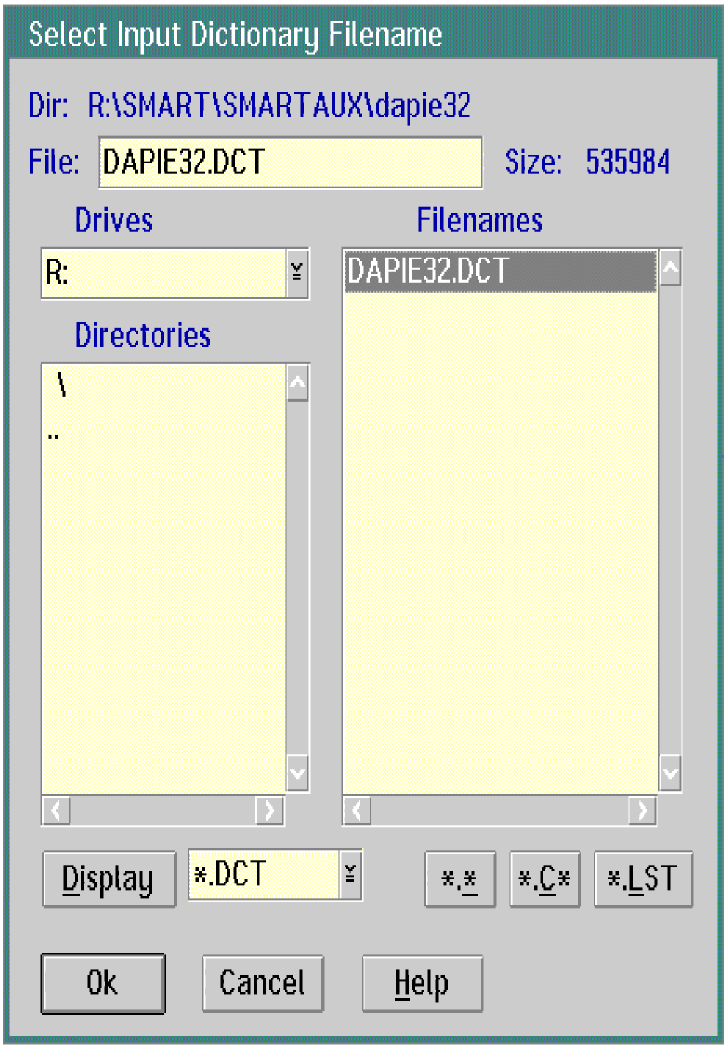
New Path: R:\SMART\SMARTAUX\dapie32

6. SMART returns to the Create User Defined Migration Data Base dialog. Click on the **Select2** push button.

7. SMART displays the Select Input Dictionary Filename dialog. In the File: entry field, type the dictionary filename:

   ```
   DAPIE32.DCT
   ```

   Click on the **Ok** push button.


Select Input Dictionary Filename

Dir: R:\SMART\SMARTAUX\dapie32

File: DAPIE32.DCT

Size: 535984

Drives: R:

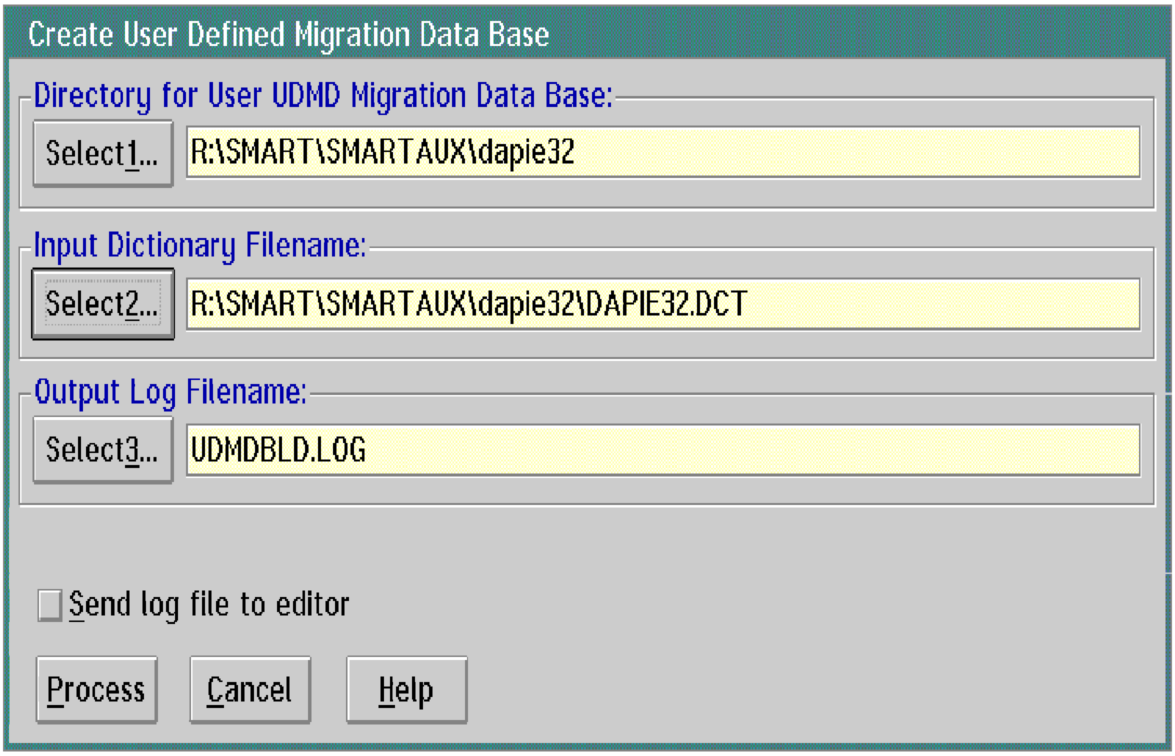
Directories:

\

..

Filenames: DAPIE32.DCT

8. SMART displays the Create User Defined Migration Data Base dialog. Click on the **Process** push button.



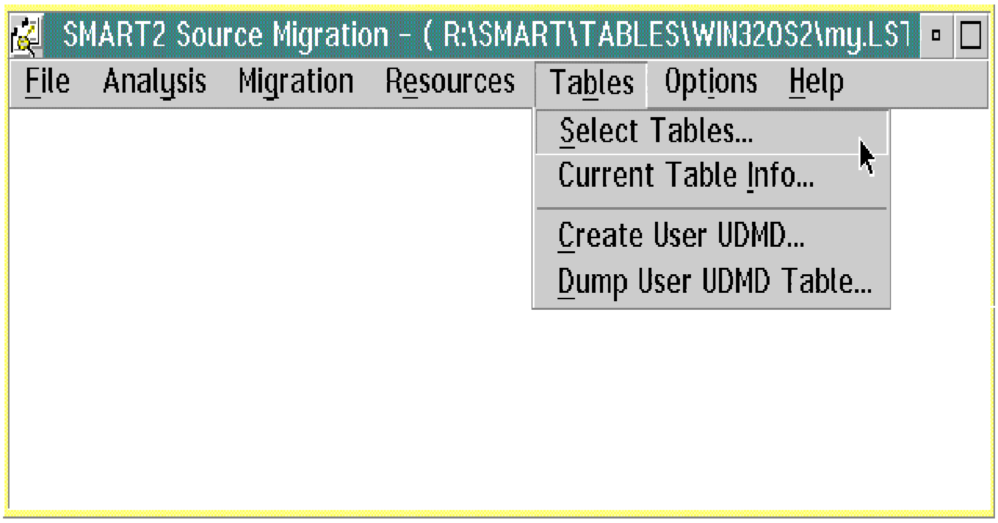
Directory for User UDMD Migration Data Base: R:\SMART\SMARTAUX\dapie32

Input Dictionary Filename: R:\SMART\SMARTAUX\dapie32\DAPIE32.DCT

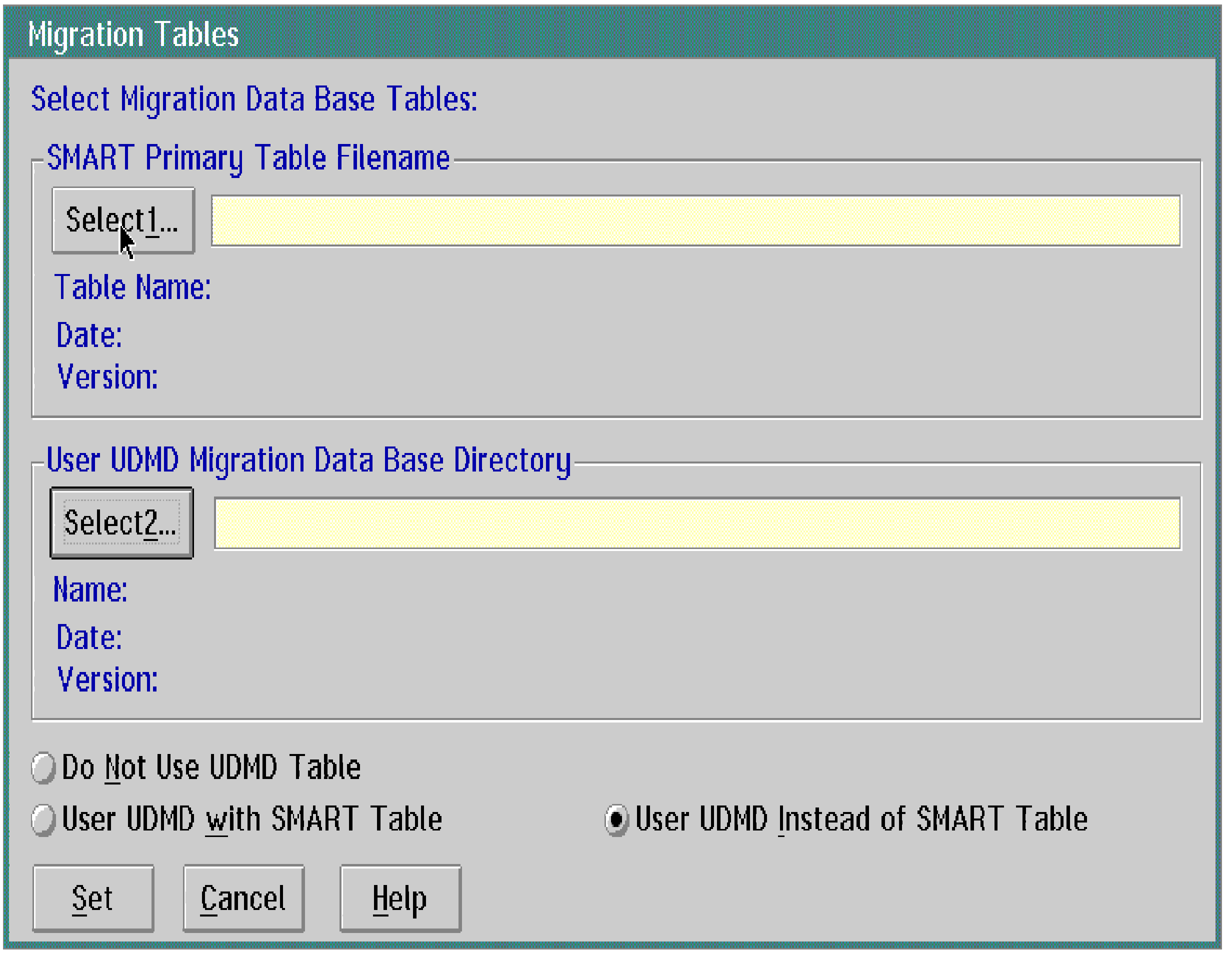
Output Log Filename: UDMDBLD.LOG

□ Send log file to editor

9. On the SMART2 Source Migration window. To load the correct table and Open32 UDMD, select the Tables pull-down. Select the Select Tables pull-down.



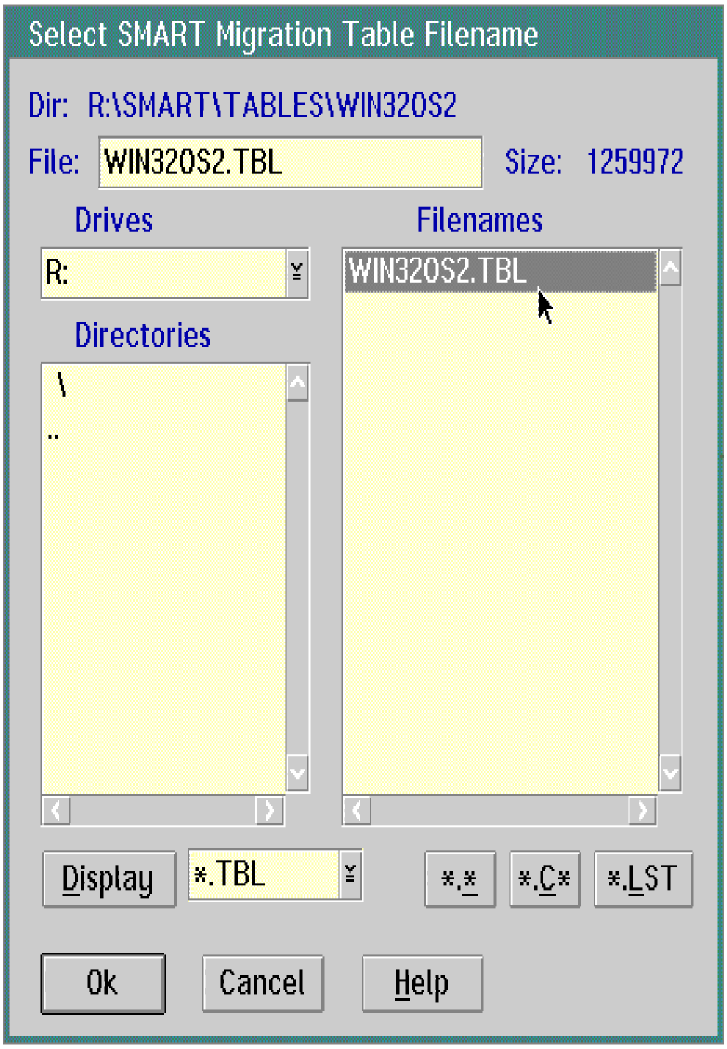
10. SMART displays the Migrations Tables dialog. Click on the **Select1** push button to select a table.


Migration Tables
Select Migration Data Base Tables:
SMART Primary Table Filename
Select1...
Table Name:
Date:
Version:
User UDMD Migration Data Base Directory
Select2...
Name:
Date:
Version:
Set
Cancel
Help

○ Do Not Use UDMD Table
○ User UDMD with SMART Table
● User UDMD Instead of SMART Table

11. SMART displays the Select SMART Migration Table Filename dialog.

   a. Go to the SMART\TABLES\WIN32OS2 subdirectory.
   b. Choose WIN32OS2.TBL.
   c. Click on the **Ok** push button.



Select SMART Migration Table Filename

Dir: R:\SMART\TABLES\WIN32OS2

File: WIN32OS2.TBL Size: 1259972

Drives: R:

Directories: \ ..

Filenames: WIN32OS2.TBL

*.TBL

12. SMART displays the Migration Tables dialog.

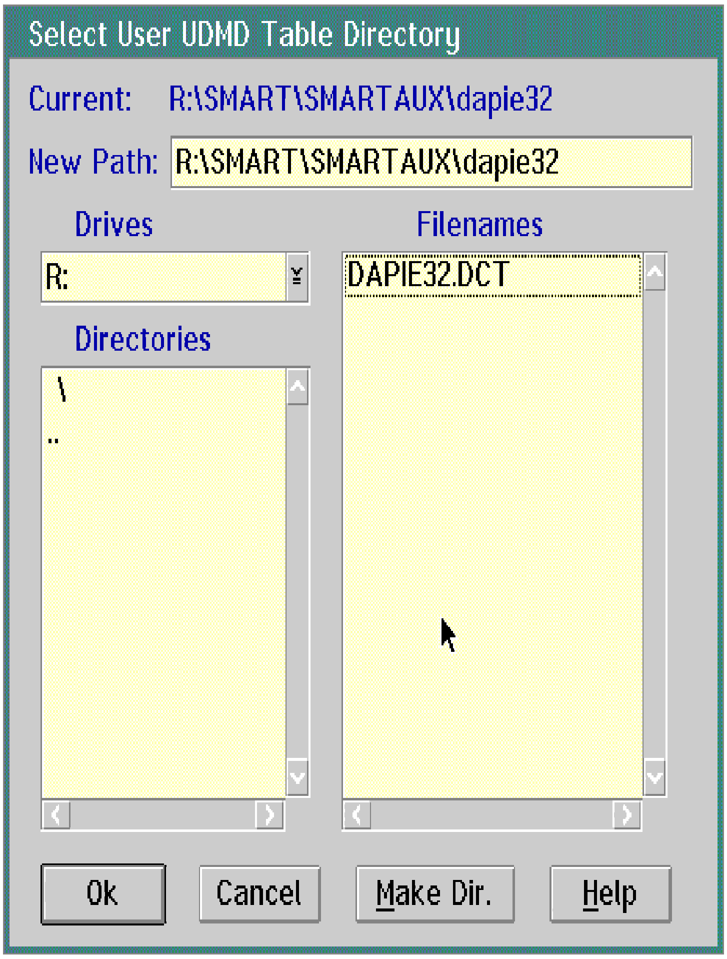
    Click the User UDMD with SMART Table radio button.

    Click on the **Select2** push button to select a UDMD.

13. SMART displays the Select User UDMD Table Directory dialog.

    Choose the SMART\SMARTAUX\DAPIE32 subdirectory.

    Click on the **Ok** push button.



Current: R:\SMART\SMARTAUX\dapie32

New Path: R:\SMART\SMARTAUX\dapie32

14. SMART displays the Migration Tables dialog.

    Click on the **Set** push button to load the table and UDMD. When the Please Standby...Loading Migration Table message disappears, SMART displays the SMART2 Source Migration window.

Analyzing Windows Code and Resource Files

You can use SMART to analyze your Windows code and resource files to determine how much effort is involved to migrate to OS/2 code and resource files. See:

- Telling SMART Where the Source Files Are
- Analyzing Windows Code
- Analyzing Windows Resource Files

Starting SMART

1. Double-click on the *SMART2 Toolset* icon. SMART displays the *SMART2 - Icon View* window, shown in step Creating the Open32 UDMD.

2. Double-click on the *SMART2 Source Migration* icon, shown in step Creating the Open32 UDMD.

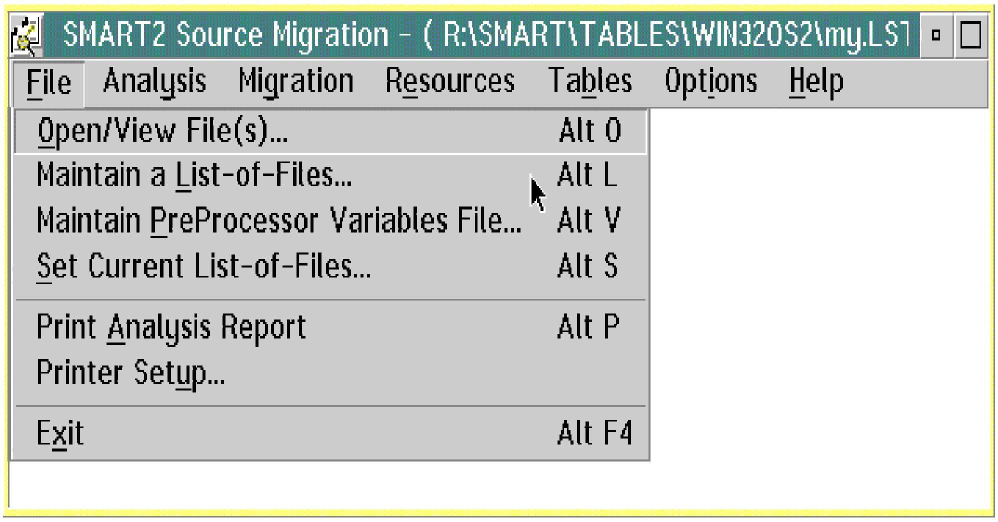
Telling SMART Where the Source Files Are

SMART requires that you build files with an extension of LST to indicate the names of your source code and header files.

Note: If you already have compiler listings, they will probably have LST extensions. Make sure you do not name the LST file you use for SMART with the same name you have used as a compiler listing: When you recompile, the SMART LST file will be overwritten.

The following instructions tell you how to create a LST file for the first time or indicate to SMART which LST file to use for analysis:

1. On the SMART2 Source Migration window, go to the File pull-down. Select Maintain a List-of-Files.

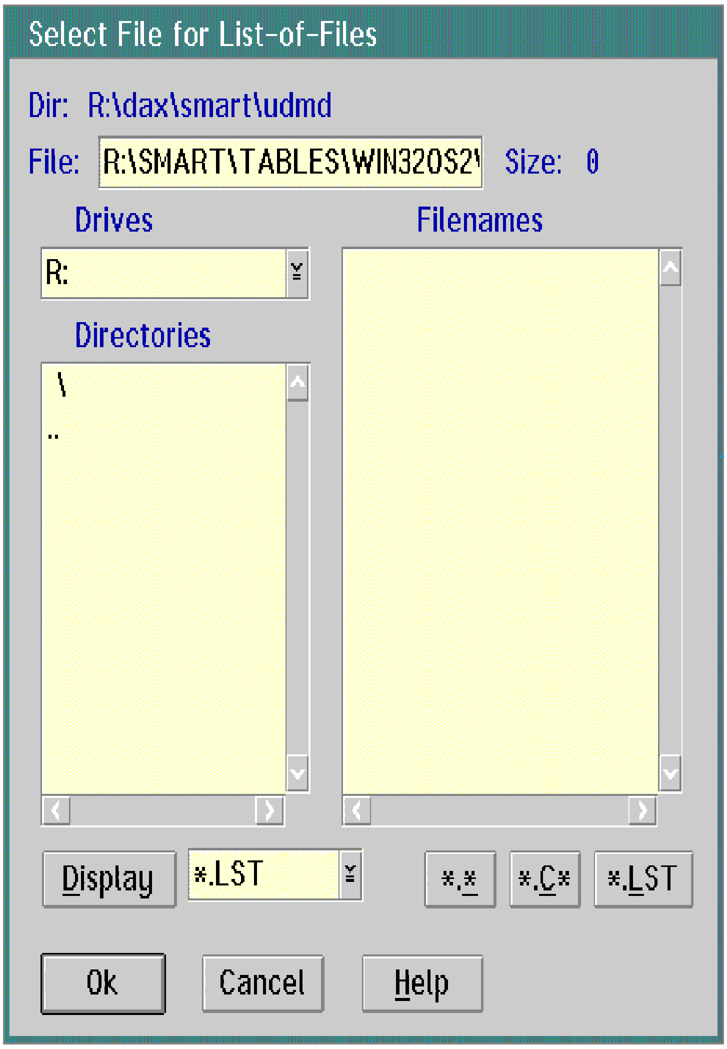

SMART2 Source Migration - (R:\SMART\TABLES\WIN32OS2\my.LST
File
Analysis
Migration
Resources
Tables
Options
Help
Open/View File(s)... Alt O
Maintain a List-of-Files... Alt L
Maintain PreProcessor Variables File... Alt V
Set Current List-of-Files... Alt S
Print Analysis Report Alt P
Printer Setup...
Exit Alt F4

2. SMART displays the Select File for List-of-Files dialog.

   a. If you already have a LST file you want to use, simply select it and click on the **Ok** push button. Then go to Analyzing Windows Code.

   b. If you have not created a LST file, create one from this dialog:

      1. In the File entry field, type the name of the LST file you wish to create.
      2. Click on the **Ok** push button.

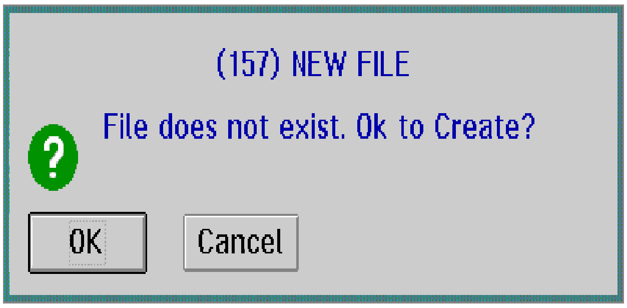


Select File for List-of-Files

Dir: R:\dax\smart\udmd

File: R:\SMART\TABLES\WIN320S2\ Size: 0

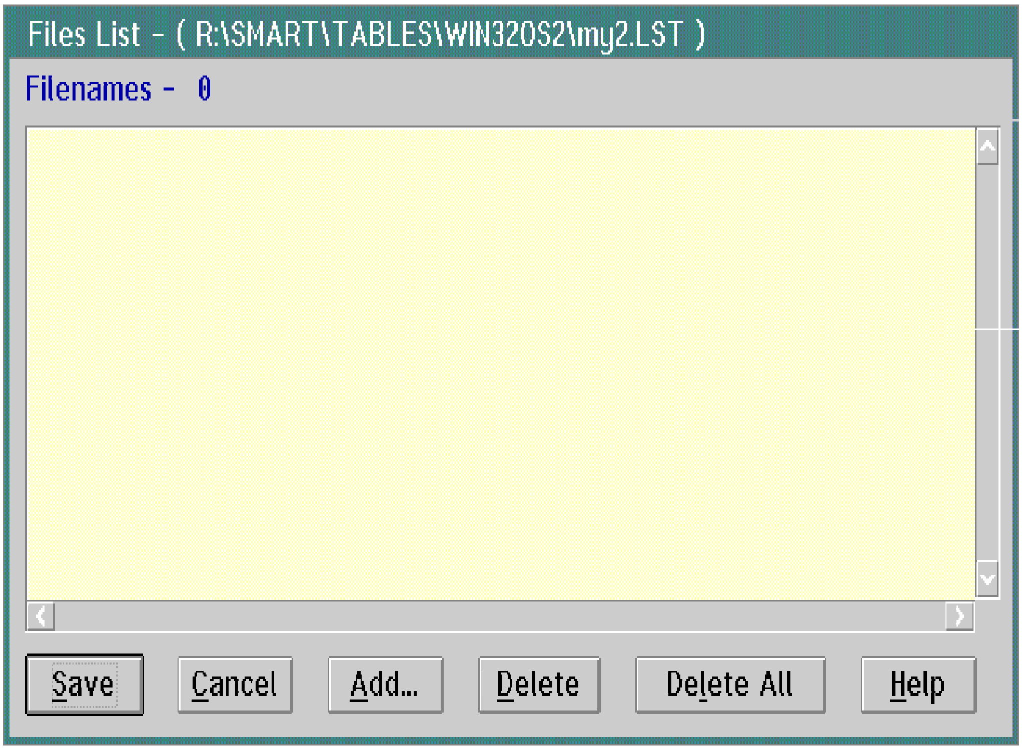
3. SMART displays the following message box. Click on the **OK** push button.



(157) NEW FILE

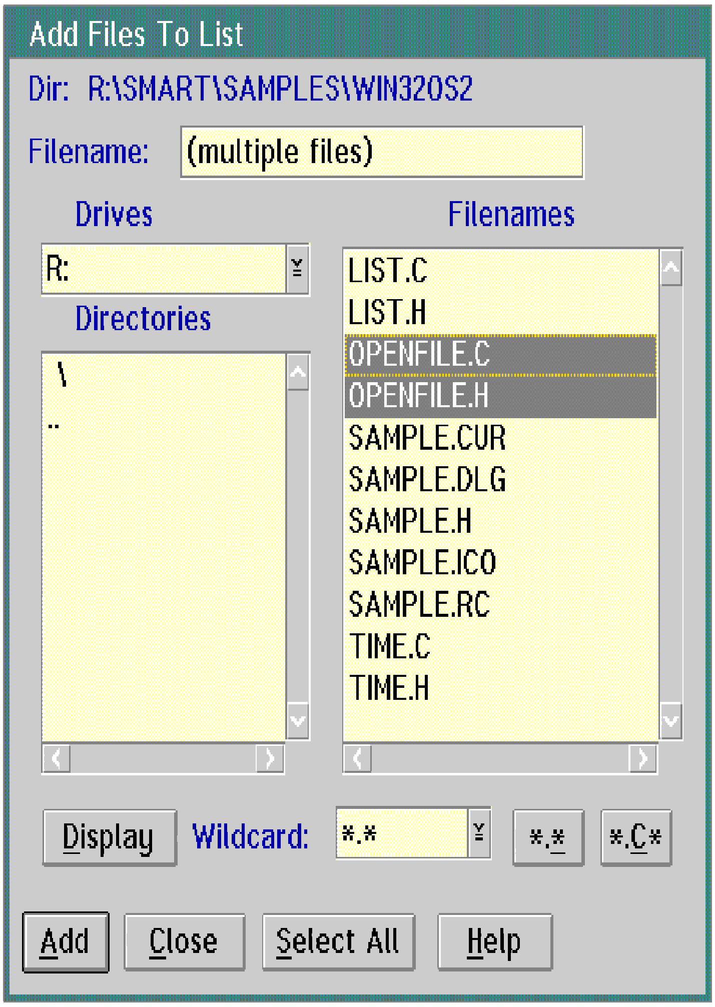
File does not exist. Ok to Create?

4. SMART displays the Files List dialog. Click on the **Add** push button.



5. SMART displays the Add Files To List dialog.

   a. Select the files you want to analyze from the Filenames list box.
   b. Click on the **Add** push button.
   c. When you have finished adding files to the LST file, click on the **Close** push button.



Dir: R:\SMART\SAMPLES\WIN32OS2

Filename: (multiple files)

R:

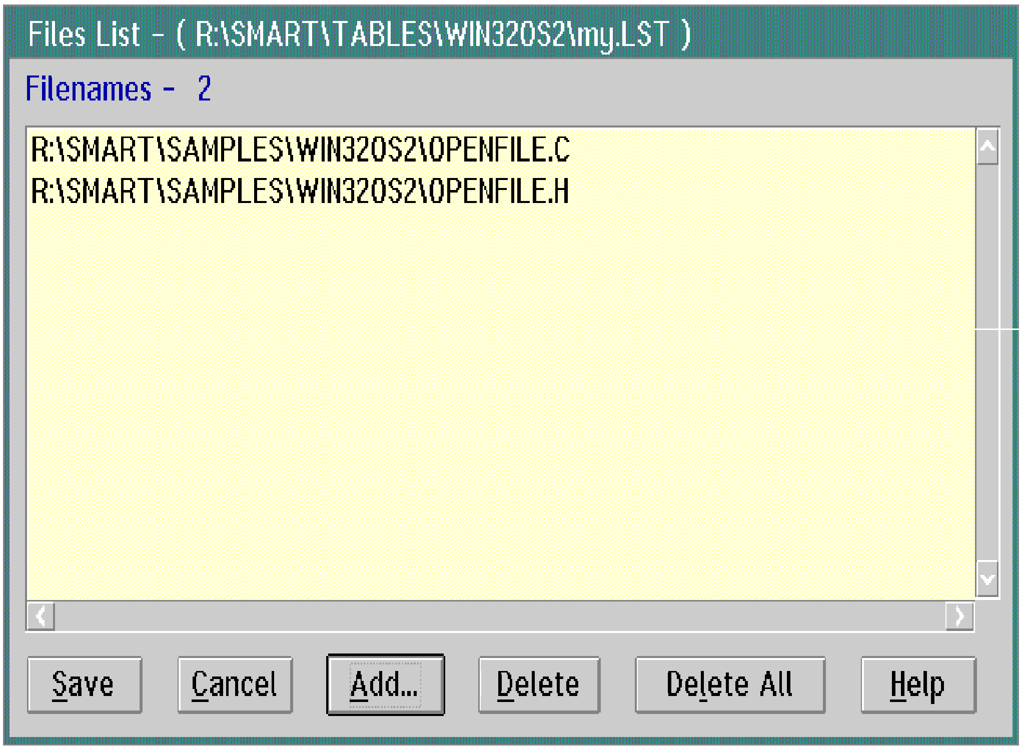
\
..

LIST.C
LIST.H
OPENFILE.C
OPENFILE.H
SAMPLE.CUR
SAMPLE.DLG
SAMPLE.H
SAMPLE.ICO
SAMPLE.RC
TIME.C
TIME.H

Wildcard: *.*

6. SMART displays the Files List dialog.

Click on the **Save** push button to save the LST file you created.

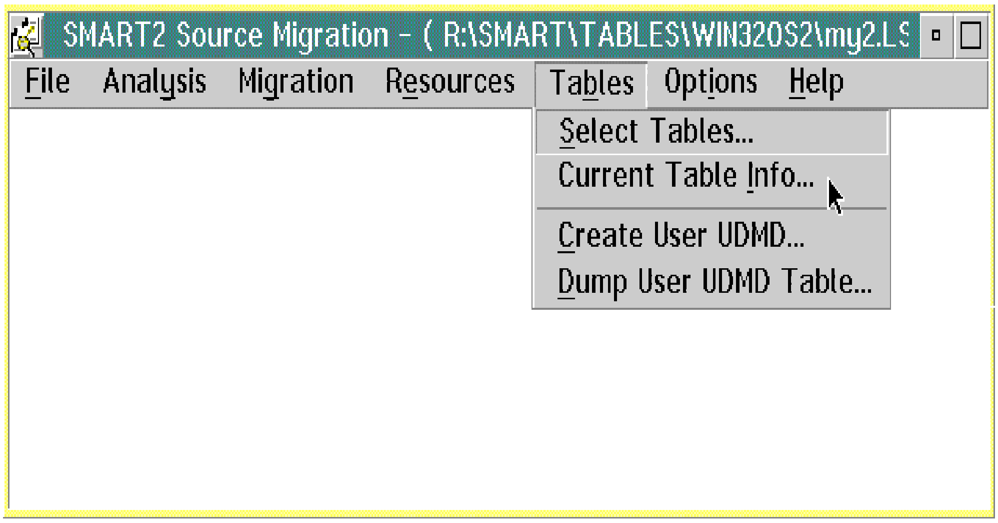


R:\SMART\SAMPLES\WIN32OS2\OPENFILE.C
R:\SMART\SAMPLES\WIN32OS2\OPENFILE.H

Analyzing Windows Code

Before analyzing source code, ensure the correct tables are loaded:

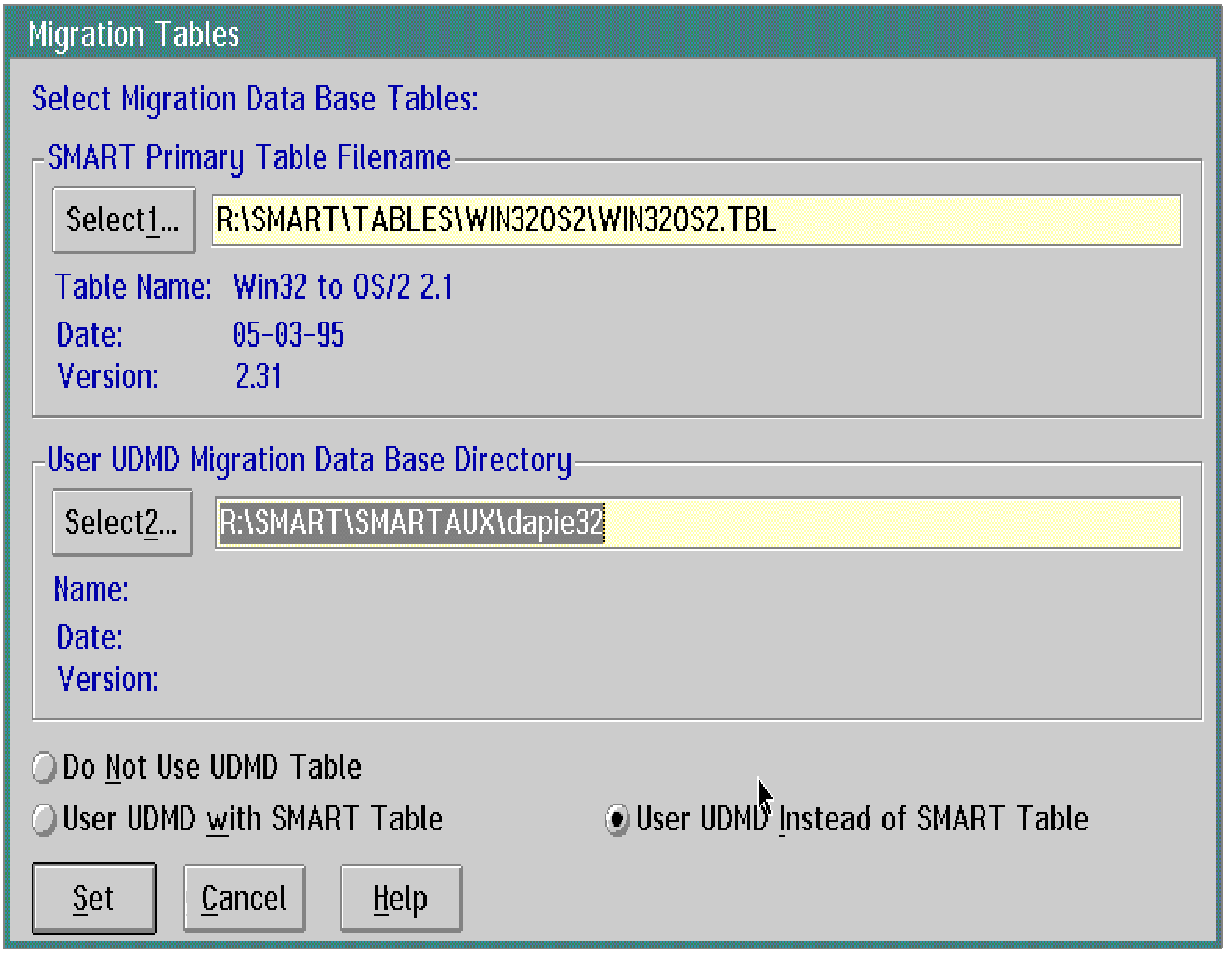
1. Select the *Tables* pull-down from *SMART2 Source Migration* window. Select the *Current Table Info...* pull-down.



2. SMART displays the *Current Table Information* informational dialog.

   The *SMART Filename* in the *SMART Table Filename* group box should be WIN32OS2.TBL. The *User UDMD Directory* in the *User UDMD Migration Data Base Directory* group box should be SMART\SMARTAUX\DAPIE32.

   a. If both are correct, click on the *Ok* push button and continue.
   b. If you need to change these fields, follow step Creating the Open32 UDMD.



Migration Tables

Select Migration Data Base Tables:

SMART Primary Table Filename

Select1... R:\SMART\TABLES\WIN32OS2\WIN32OS2.TBL

Table Name: Win32 to OS/2 2.1

Date: 05-03-95

Version: 2.31

User UDMD Migration Data Base Directory

Select2... R:\SMART\SMARTAUX\dapie32

Name:

Date:

Version:

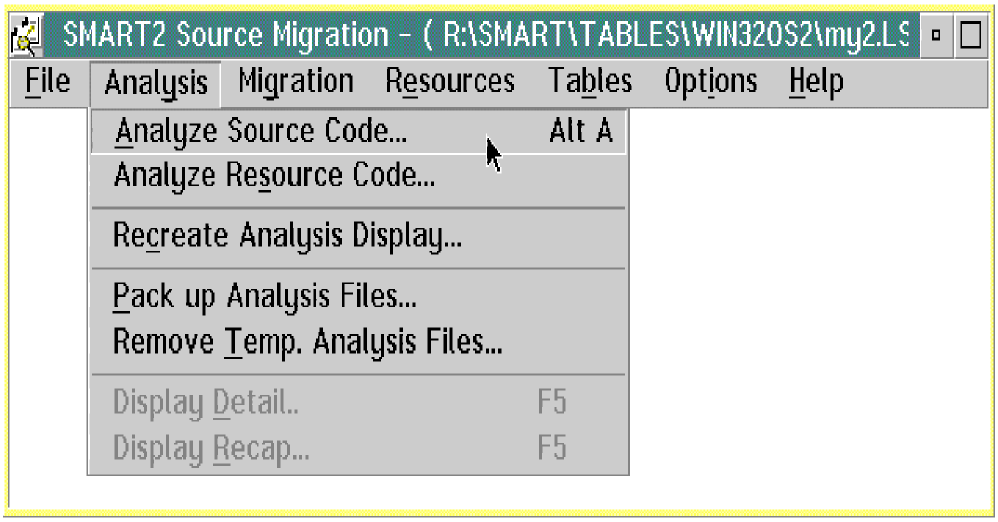
○ Do Not Use UDMD Table

○ User UDMD with SMART Table

● User UDMD Instead of SMART Table

3. SMART displays the SMART2 Source Migration window.

   a. Go to the Analysis pull-down.
   b. Select Analyze Source Code.



SMART2 Source Migration - (R:\SMART\TABLES\WIN32OS2\my2.LS

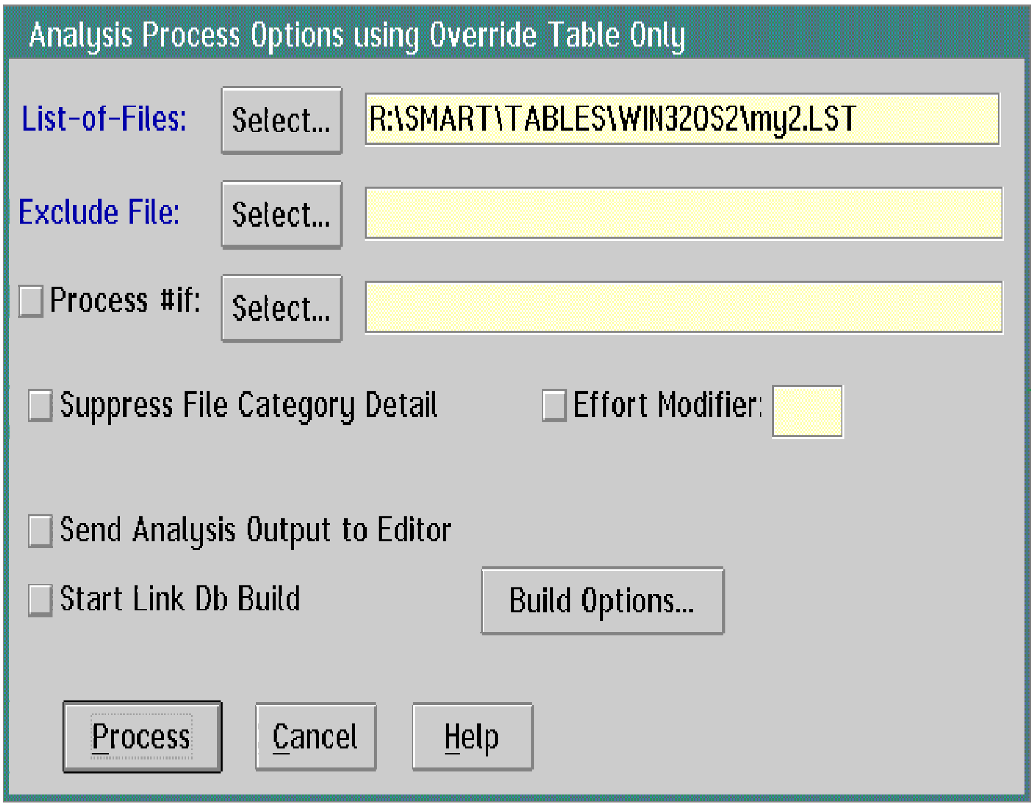
File
Analysis
Migration
Resources
Tables
Options
Help
Analyze Source Code... Alt A
Analyze Resource Code...
Recreate Analysis Display...
Pack up Analysis Files...
Remove Temp. Analysis Files...
Display Detail.. F5
Display Recap... F5

4. SMART displays the Analysis Process Options using Override Table Only dialog. In this dialog, SMART automatically chooses the list of files you created previously. (See Telling SMART Where the Source Files Are.) Either:

- Leave the defaults in this window
- Select a different LST file by either:
  a. Typing a different LST file in the List-of-Files entry field using the full path name

b. Click on the **Select** push button next to the List-of-Files entry field to see a dialog for selecting a new LST file.

Click on the **Process** push button.



Analysis Process Options using Override Table Only

List-of-Files: Select... R:\SMART\TABLES\WIN32OS2\my2.LST

Exclude File: Select...

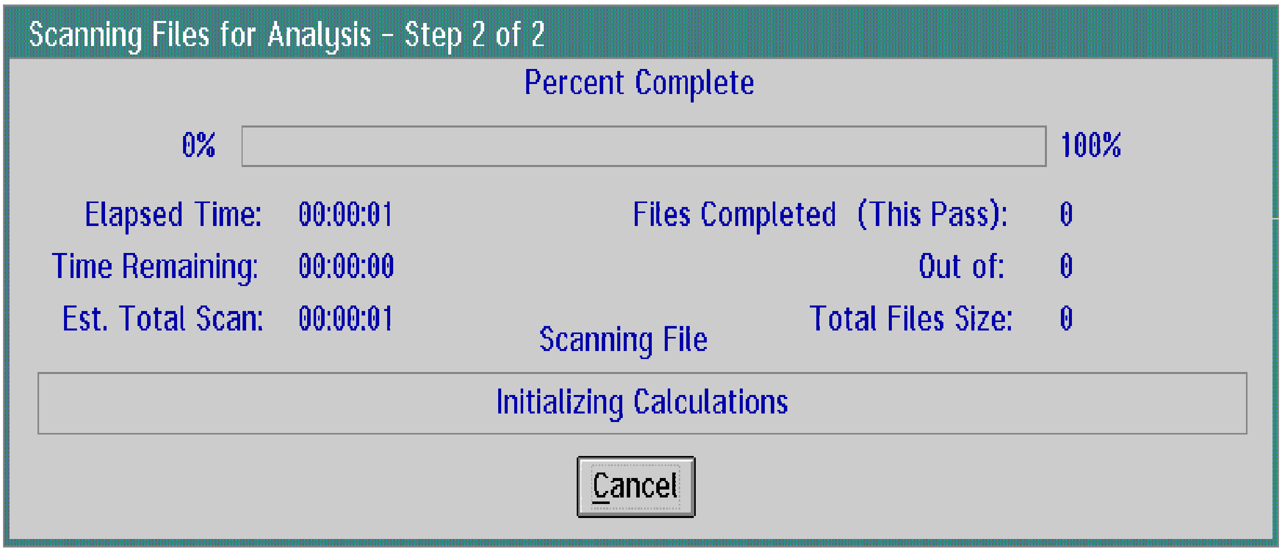
□ Process #if: Select...

□ Suppress File Category Detail □ Effort Modifier:

□ Send Analysis Output to Editor

□ Start Link Db Build

5. SMART displays a progress informational dialog to show the progress of the analysis.



Scanning Files for Analysis - Step 2 of 2

Percent Complete

0% 100%

Elapsed Time: 00:00:01 Files Completed (This Pass): 0

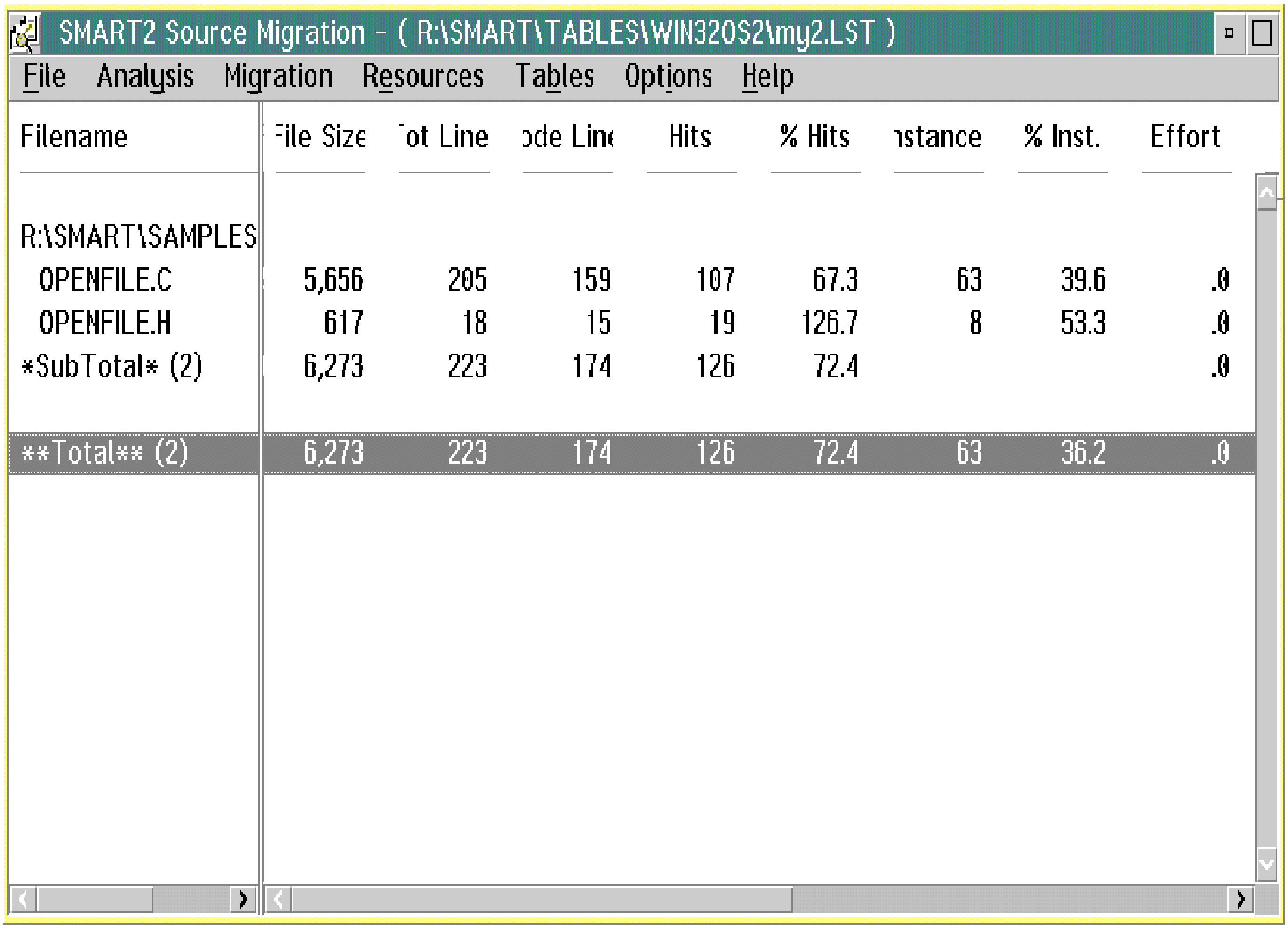
Time Remaining: 00:00:00 Out of: 0

Est. Total Scan: 00:00:01 Total Files Size: 0

Scanning File

Initializing Calculations

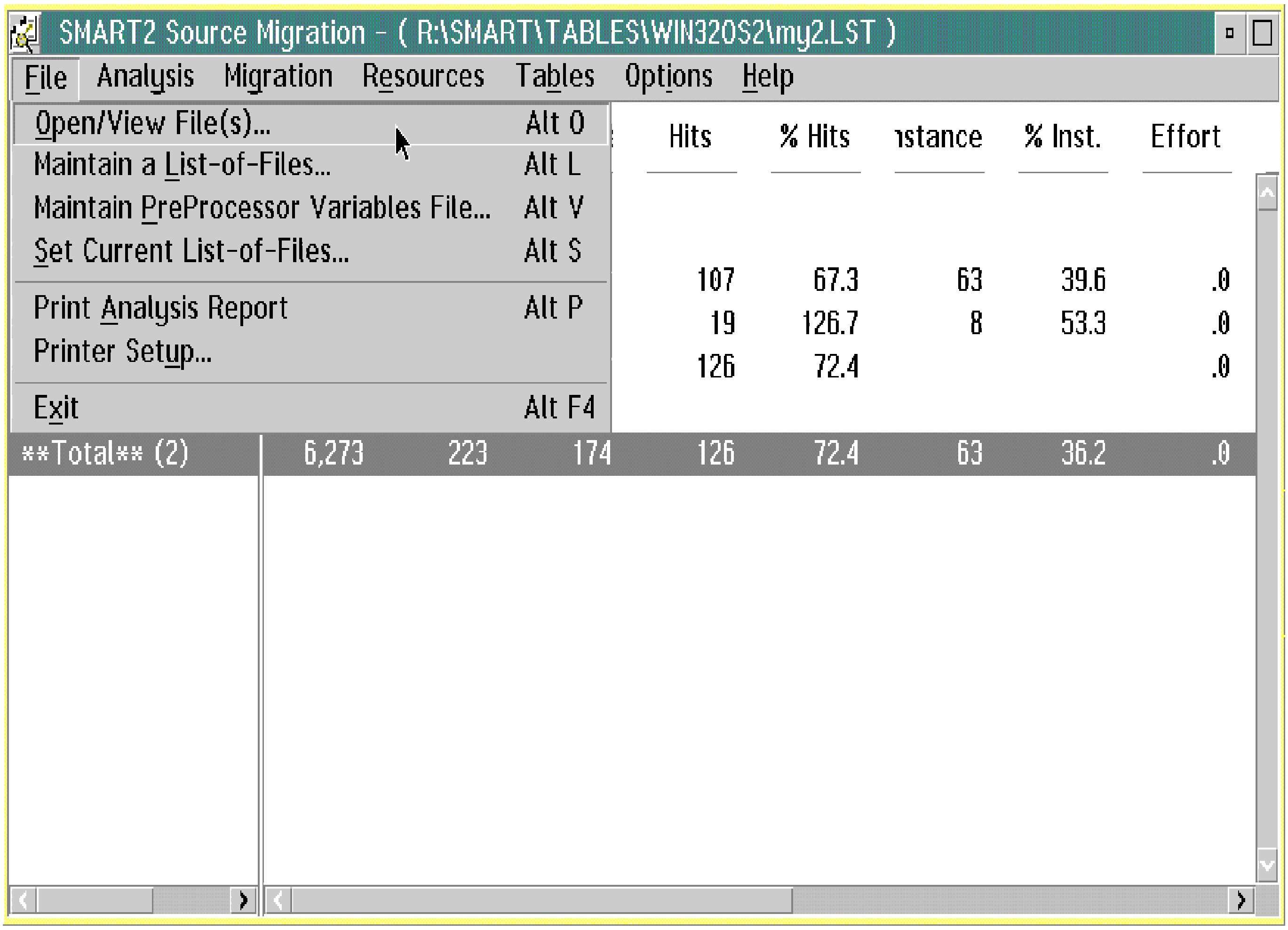
6. SMART displays an initial analysis report in the SMART2 Source Migration window.



SMART2 Source Migration - (R:\SMART\TABLES\WIN32OS2\my2.LST)
File Analysis Migration Resources Tables Options Help

Filename	File Size	Tot Line	ode Line	Hits	% Hits	nstance	% Inst.	Effort
R:\SMART\SAMPLES								
OPENFILE.C	5,656	205	159	107	67.3	63	39.6	.0
OPENFILE.H	617	18	15	19	126.7	8	53.3	.0
SubTotal (2)	6,273	223	174	126	72.4			.0
Total (2)	6,273	223	174	126	72.4	63	36.2	.0

7. To see other analysis reports, go to the File pull-down and choose Open/View File(s)


SMART2 Source Migration - (R:\SMART\TABLES\WIN32OS2\my2.LST)
File
Analysis
Migration
Resources
Tables
Options
Help
Open/View File(s)... Alt O
Maintain a List-of-Files... Alt L
Maintain PreProcessor Variables File... Alt V
Set Current List-of-Files... Alt S
Print Analysis Report Alt P
Printer Setup...
Exit Alt F4
Hits
% Hits
ıstance
% Inst.
Effort
107 67.3 63 39.6 .0
19 126.7 8 53.3 .0
126 72.4 .0
Total (2) 6,273 223 174 126 72.4 63 36.2 .0

8. The file with the extension RPT is the code analysis report. Select this report. Click on the **Open** push button to view the report.

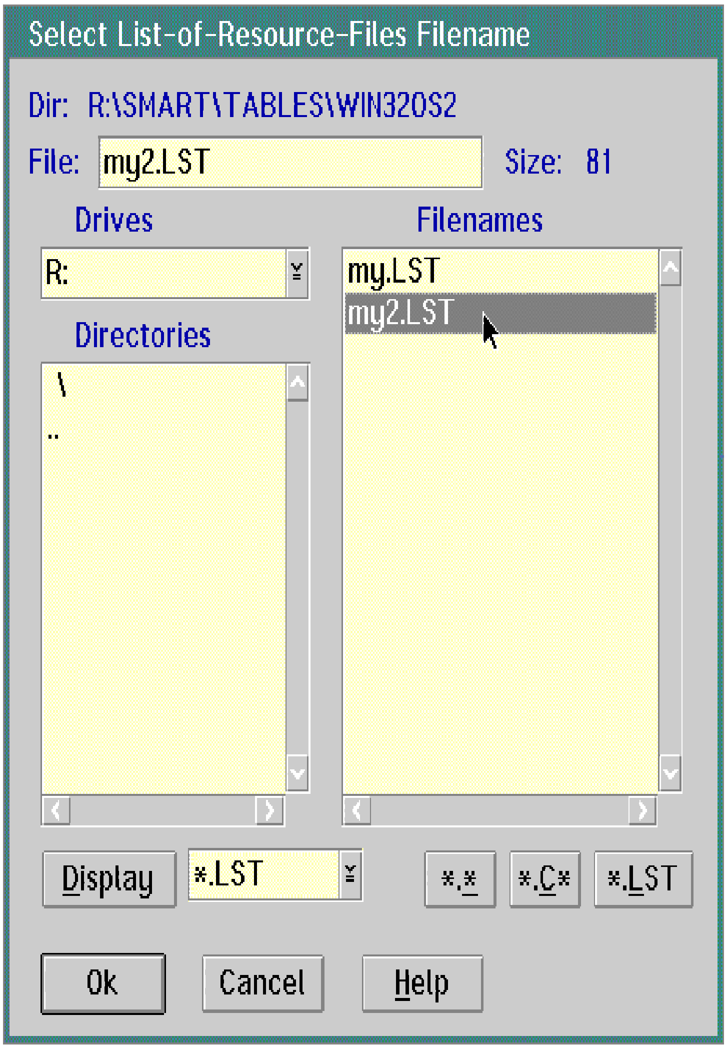
9. SMART displays the report in the SLink - The Smart Editor window. Use the Help pull-down on this window to get information about reports.

Analyzing Windows Resource Files

1. On the SMART2 Source Migration window, go to the Analysis pull-down. Select Analyze Resource Code.

2. SMART displays the Select File for a List-of-Resource-Files dialog. In this dialog, SMART2 automatically chooses the list of files you created previously. (See Telling SMART Where the Source Files Are.)

   a. Select the list (LST) of files that contains the RC files to analyze.
   b. Click on the **Ok** push button.


Select List-of-Resource-Files Filename

Dir: R:\SMART\TABLES\WIN32OS2

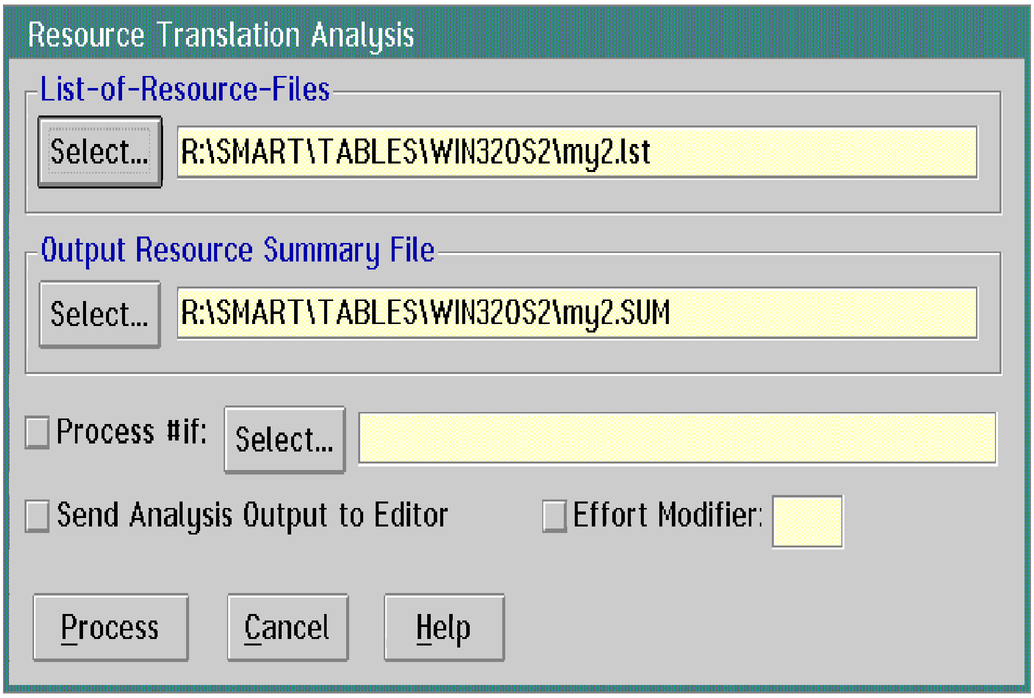
File: my2.LST Size: 81

Drives: R:

Directories:
\
..

Filenames:
my.LST
my2.LST

3. SMART displays the Resource Translation Analysis dialog. Click on the **Process** push button.


Resource Translation Analysis

List-of-Resource-Files: R:\SMART\TABLES\WIN32OS2\my2.lst

Output Resource Summary File: R:\SMART\TABLES\WIN32OS2\my2.SUM

□ Process #if:

□ Send Analysis Output to Editor

□ Effort Modifier:

4. SMART displays a progress informational dialog to show the progress of the analysis.

5. SMART returns you to the SMART2 Source Migration window. To see the resource analysis report:

   a. Go to the File pull-down.

   b. Choose Open/View File(s)

   c. The file with the extension SUM is the resource analysis report. Select SUM report.

   d. Click on the **Open** push button to view the report.

   e. SMART displays the report in the SLink - The Smart Editor window. Use the Help pull-down on this window to get information about this report.

Migrating Windows Code, Resource Files, and Help Files

You can use SMART to migrate your Windows code, resource and help files to code, resource and help files that will compile in both Windows and OS/2. You can also use Hyperwise Lite to migrate Windows 32-bit help files to OS/2 help files. See the documentation that accompanies the Hyperwise Lite tool.

To use SMART, see:

- Migrating Windows Code
- Migrating Windows Resource Files
- Migrating Windows Help Files

Starting SMART

1. Double-click on the *SMART2 Toolset* icon. SMART displays the *SMART2 - Icon View* window, shown in step Creating the Open32 UDMD.

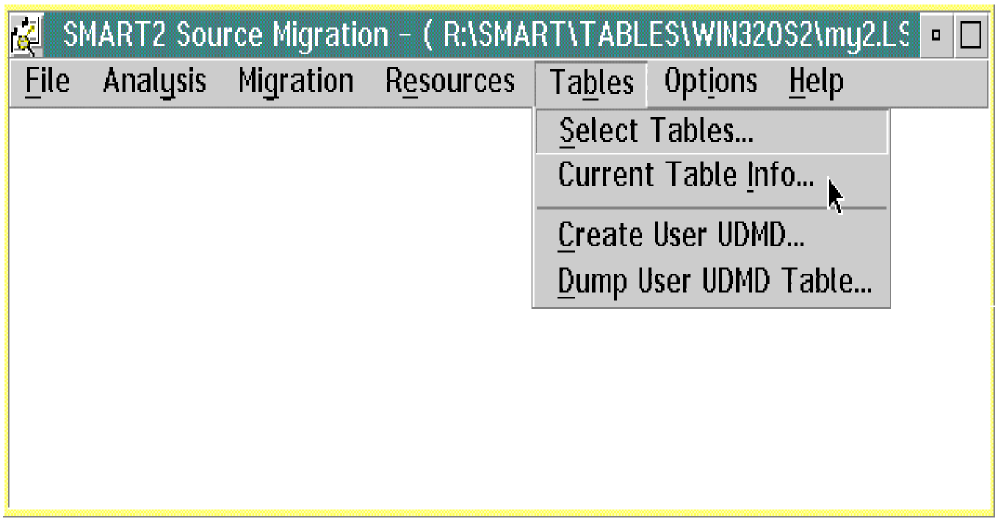
2. Double-click on the *SMART2 Source Migration* icon, shown in step Creating the Open32 UDMD.

Migrating Windows Code

Note: These steps tell you how to migrate Windows 32-bit code only.

Before migrating source code, ensure the correct tables are loaded:

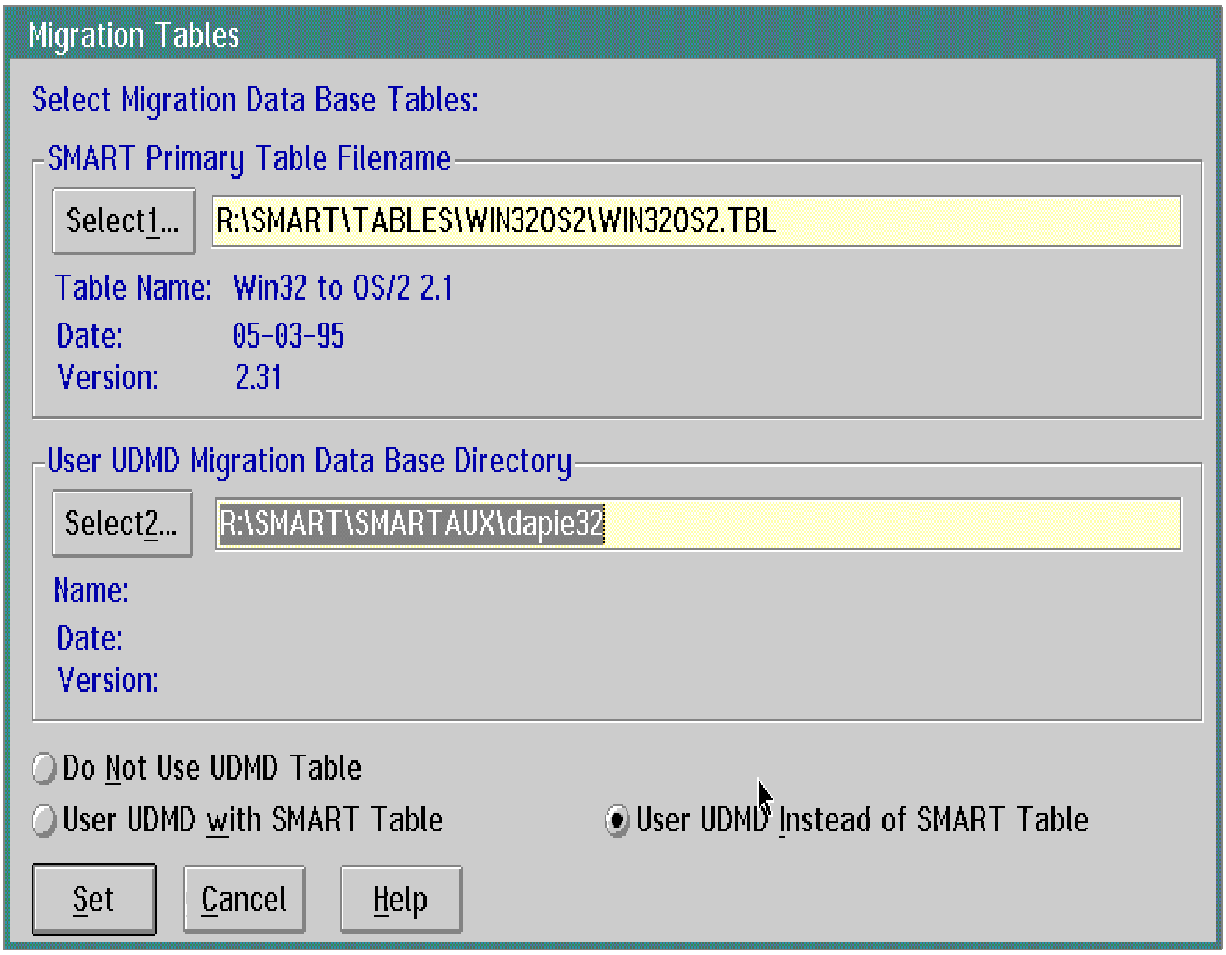
1. Select the *Tables* pull-down from *SMART2 Source Migration* window. Select the *Current Table Info...* pull-down.



2. SMART displays the *Current Table Information* informational dialog.

The *SMART Filename* in the *SMART Table Filename* group box should be WIN32OS2.TBL. The *User UDMD Directory* in the *User UDMD Migration Data Base Directory* group box should be SMART\SMARTAUX\DAPIE32.

a. If both are correct, click on the *Ok* push button and continue.
b. If you need to change these fields, follow step Creating the Open32 UDMD.



Migration Tables

Select Migration Data Base Tables:

SMART Primary Table Filename

Select1... R:\SMART\TABLES\WIN32OS2\WIN32OS2.TBL

Table Name: Win32 to OS/2 2.1
Date: 05-03-95
Version: 2.31

User UDMD Migration Data Base Directory

Select2... R:\SMART\SMARTAUX\dapie32

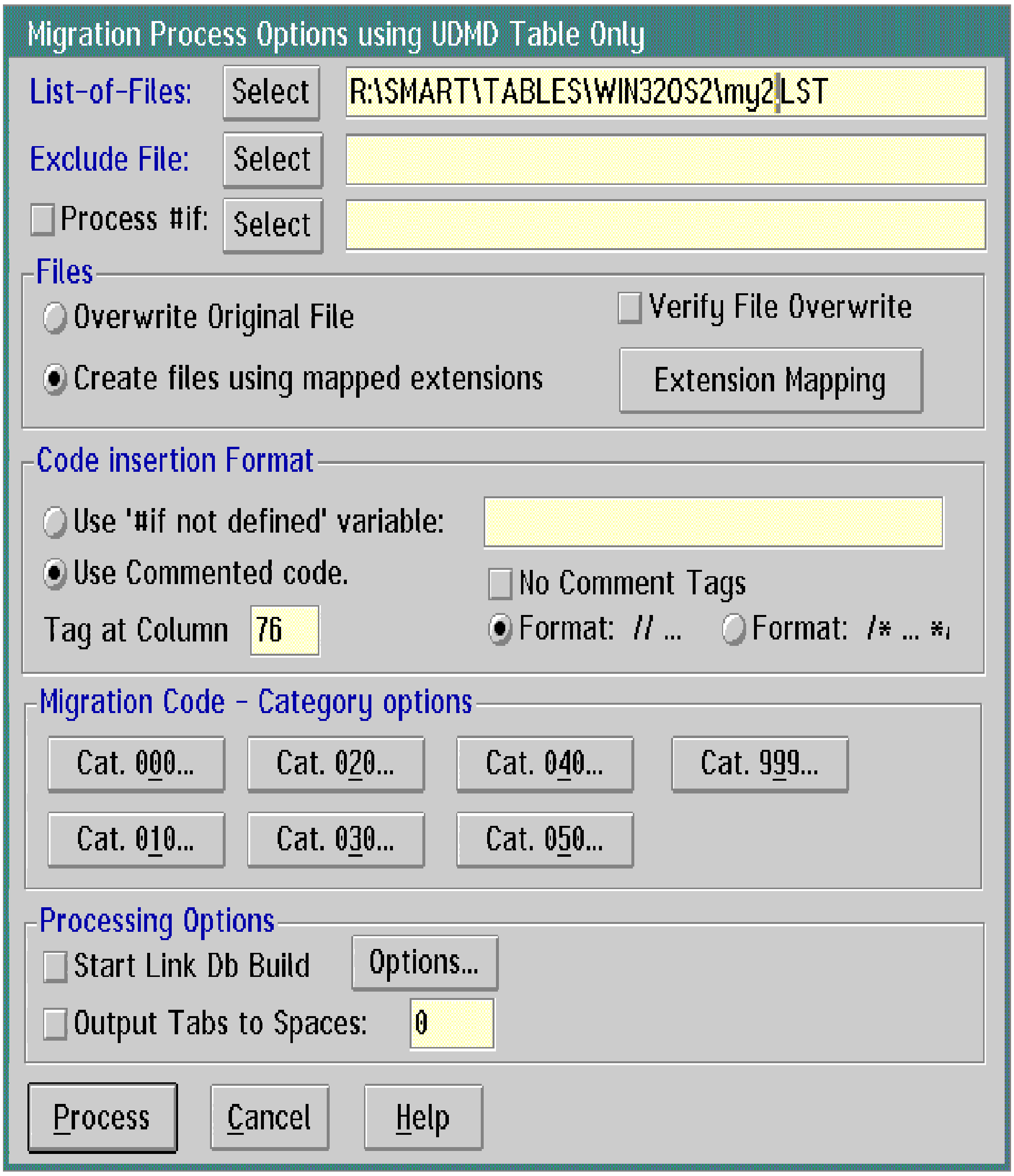
Name:
Date:
Version:

○ Do Not Use UDMD Table
○ User UDMD with SMART Table
● User UDMD Instead of SMART Table

3. On the SMART2 Source Migration window, go to the Migration pull-down. Select Migrate Source Code.

4. SMART displays the Migration Process Options using UDMD Table Only dialog. SMART automatically enters the name of the LST file you initially created in the List-of-Files entry field. You can use this file name or type in another file name. Click on the

Process push button.



Migration Process Options using UDMD Table Only

List-of-Files: Select R:\SMART\TABLES\WIN32OS2\my2.LST

Exclude File: Select

□ Process #if: Select

Files

○ Overwrite Original File

□ Verify File Overwrite

● Create files using mapped extensions

Code insertion Format

○ Use '#if not defined' variable:

● Use Commented code.

□ No Comment Tags

Tag at Column 76

● Format: // ... ○ Format: /* ... */

Migration Code - Category options

Processing Options

□ Start Link Db Build

□ Output Tabs to Spaces: 0

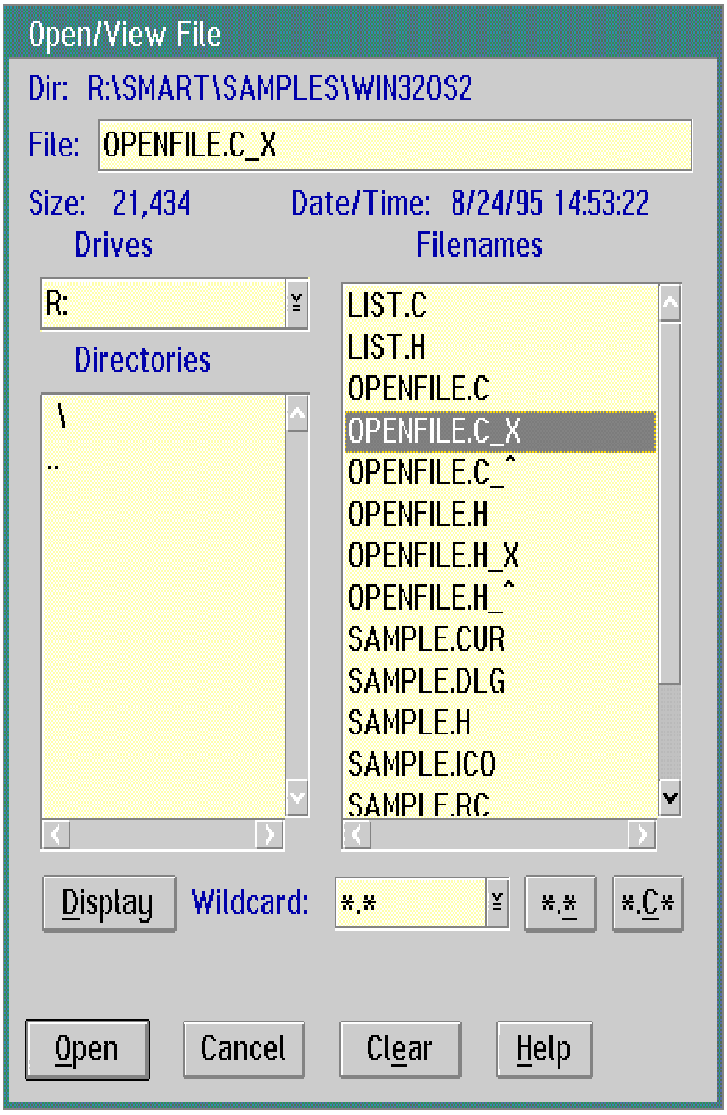
5. SMART displays the SMART2 Source Migration window.

   a. Go to the File pull-down.
   b. Select Open/View File(s).

6. SMART displays the Open/View File dialog. If you chose the overwrite original option from the Migrate Source Code option, SMART places the converted C code in the file with the C extension and places the converted header file in the file with the H extension. Otherwise, the file with the extension C_X is the converted C code and the file with the extension H_X is the converted header file.

   Select the converted files.

   Click on the **Open** push button to view the code.



Open/View File

Dir: R:\SMART\SAMPLES\WIN32OS2

File: OPENFILE.C_X

Size: 21,434 Date/Time: 8/24/95 14:53:22

Drives: R:

Directories:

```
\
..
```

Filenames:

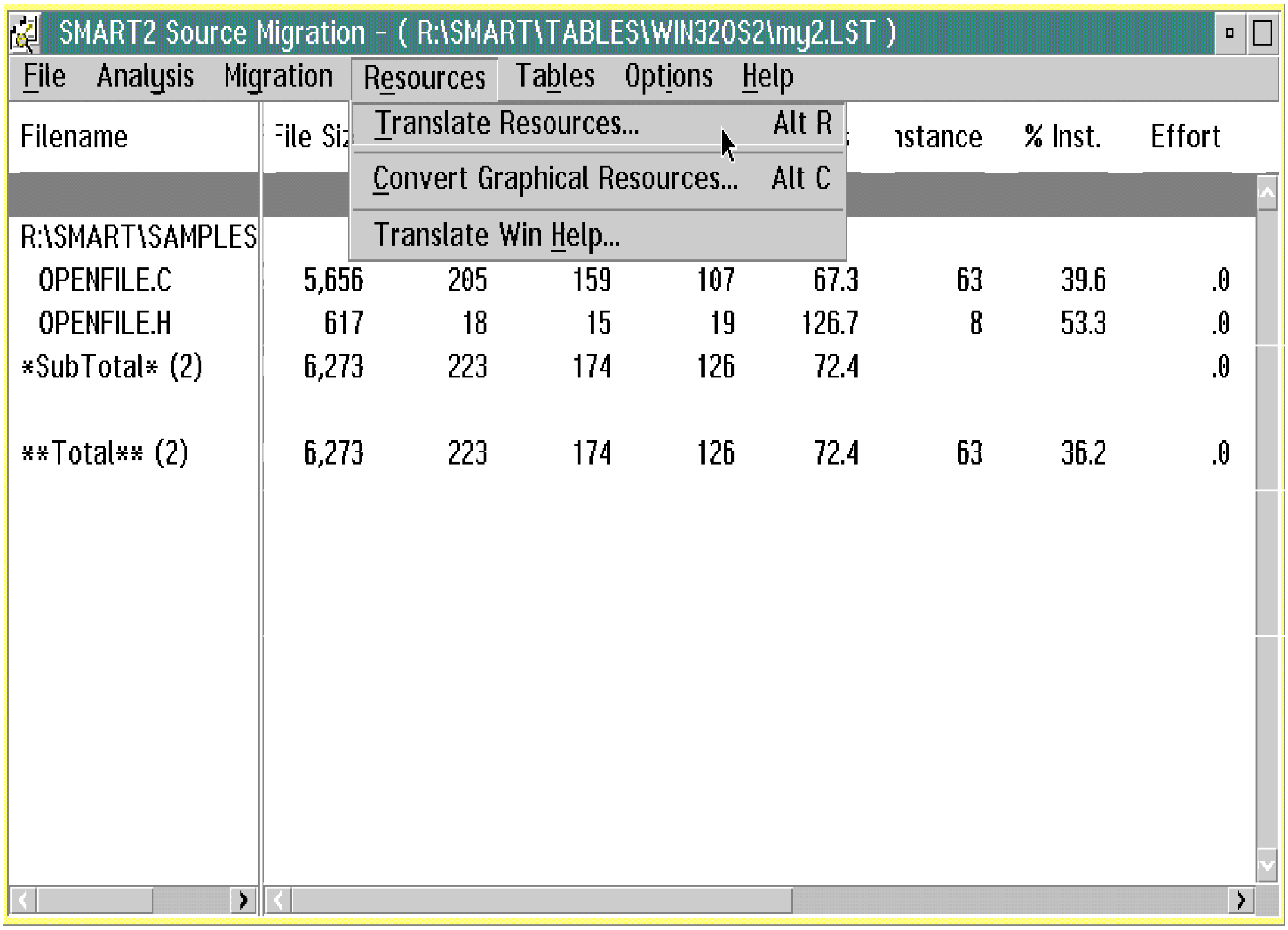
```
LIST.C
LIST.H
OPENFILE.C
OPENFILE.C_X
OPENFILE.C_^
OPENFILE.H
OPENFILE.H_X
OPENFILE.H_^
SAMPLE.CUR
SAMPLE.DLG
SAMPLE.H
SAMPLE.ICO
SAMPLE.RC
```

Wildcard: *.*

7. SMART displays the code in the SLink - The Smart Editor window. The comments in the code indicate what SMART2 did to convert your code. Use the Help pull-down on this window to get information about the migrated code.

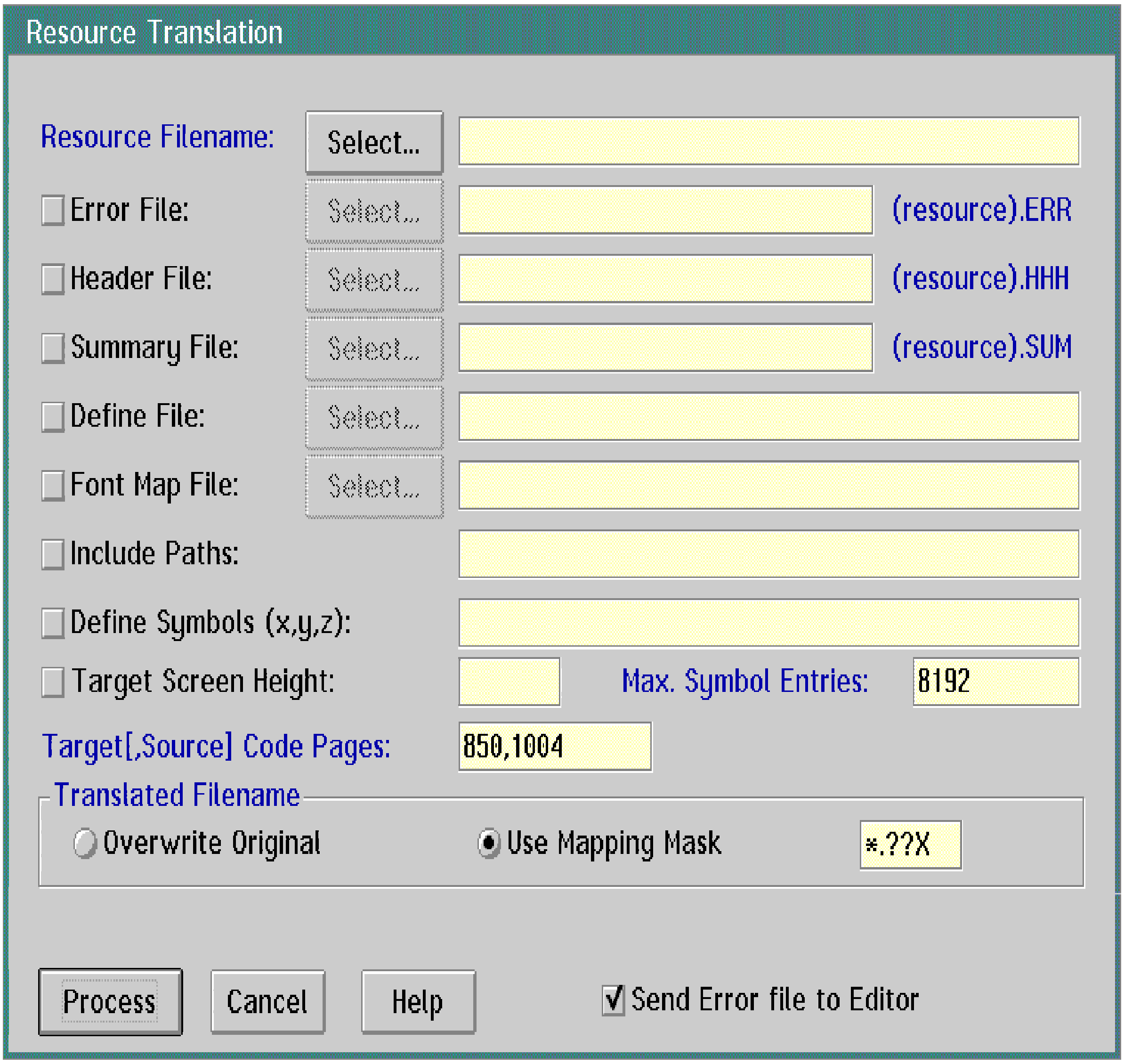
Migrating Windows Resource Files

1. On the SMART2 Source Migration window, go to the Resources pull-down. Select Translate Resources.


SMART2 Source Migration - (R:\SMART\TABLES\WIN32OS2\my2.LST)
File
Analysis
Migration
Resources
Tables
Options
Help
Translate Resources... Alt R
Convert Graphical Resources... Alt C
Translate Win Help...

Filename	File Siz					nstance	% Inst.	Effort
R:\SMART\SAMPLES								
OPENFILE.C	5,656	205	159	107	67.3	63	39.6	.0
OPENFILE.H	617	18	15	19	126.7	8	53.3	.0
SubTotal (2)	6,273	223	174	126	72.4			.0
Total (2)	6,273	223	174	126	72.4	63	36.2	.0

2. SMART displays the Resource Translation dialog. Click on the **Select** push button next to the Resource Filename: entry field.



Max. Symbol Entries: 8192

Target[,Source] Code Pages: 850,1004

● Use Mapping Mask *.??X

☑ Send Error file to Editor

3. SMART displays the Select Resource Filename dialog. Select a resource file. Check the Send error file to editor check box so that SMART logs status information. Click on the **Ok** push button.

4. SMART displays the Resource Translation dialog. Click on the **Process** push button.

5. SMART displays the SMART2 Source Migration window. Go to the File pull-down. Select Open/View File(s).

6. SMART displays the Open/View File dialog. Select the converted resource file. Click on the **Open** push button to view the resource file.

7. SMART displays the resource file in the SLink - The Smart Editor window. The comments in the resource file indicate what SMART did to convert your resource file. Use the Help pull-down on this window to get information about the migrated resource file.

Note: The Resource Translator in SMART allows the choice of converting resources in one of the following formats:

1. String id: Check the Supports String ID check box when using the Open32 functions for resources. The .hhh file is not required. The Resource Compiler accepts the resources with ids specified as quoted strings.

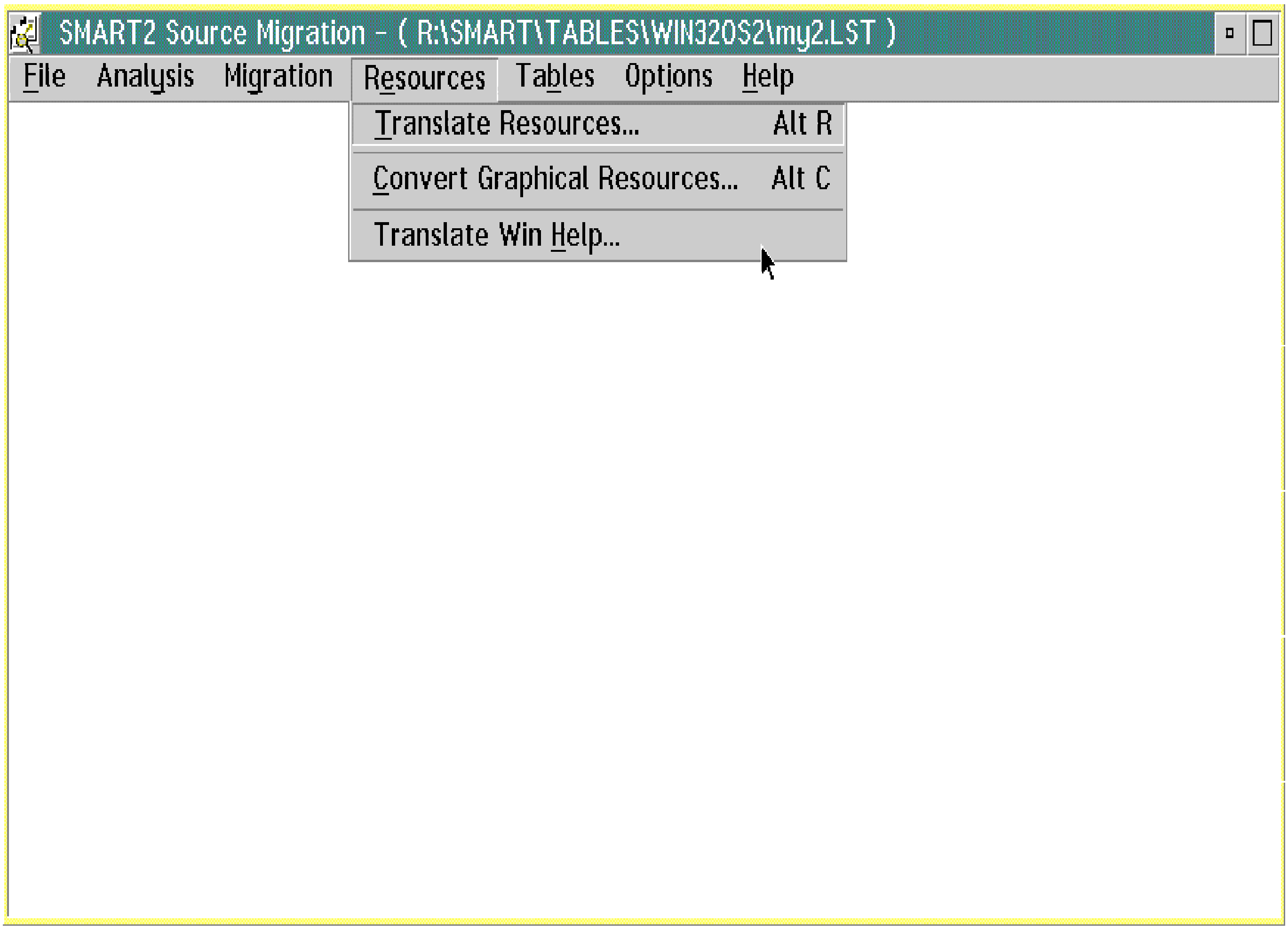
2. Integer id: Make sure the Supports String ID check box is not checked in order to generate the required .hhh file. This file defines the resource ids for use with the OS/2 Warp native functions.

Migrating Windows Help Files

You can use either SMART or Hyperwise Lite to migrate Windows 32-bit help files to OS/2 help files. See the Hyperwise Lite documentation for information on using Hyperwise Lite to migrate help files.

Follow these steps use SMART to convert a Windows rich text format (RTF) file to an OS/2 help (IPF) file:

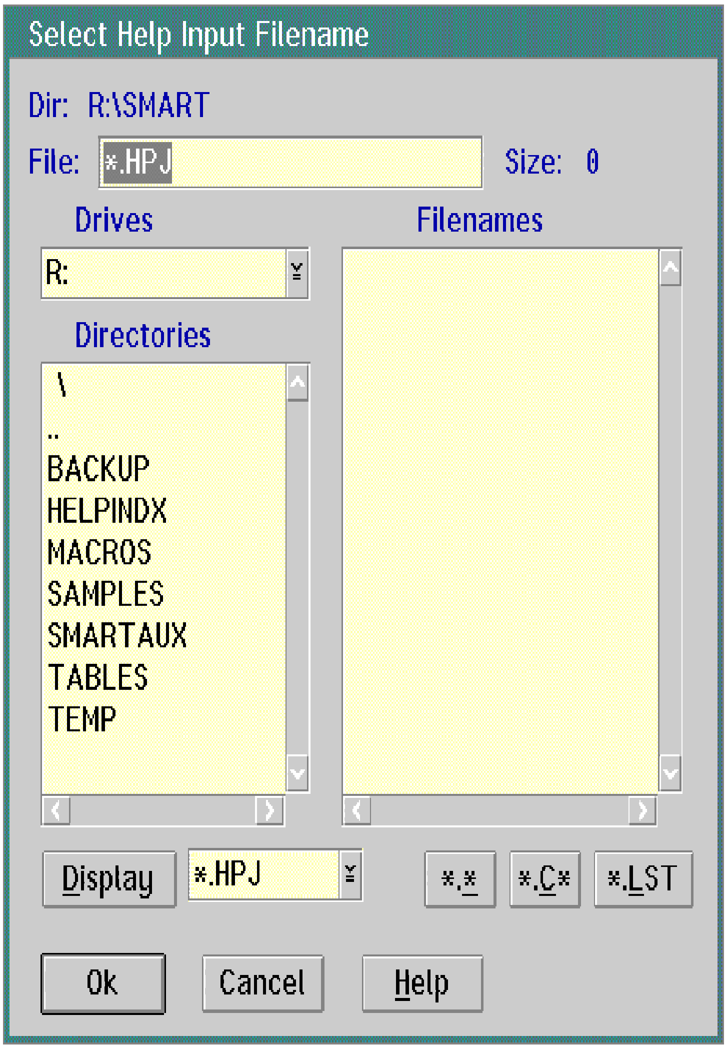
1. On the SMART2 Source Migration window, go to the Resources pull-down. Select Translate Win Help.


SMART2 Source Migration - (R:\SMART\TABLES\WIN32OS2\my2.LST)
File
Analysis
Migration
Resources
Tables
Options
Help
Translate Resources... Alt R
Convert Graphical Resources... Alt C
Translate Win Help...

2. SMART displays the Select Help Input Filename dialog. Create an HPJ file that contains the name of the RTF file to be converted along with any bitmap names that are needed.

   A sample HPJ file is:

   ```
   ..SMARTAUX\SAMPLE.HPJ
   ```



Dir: R:\SMART

File: *.HPJ

Size: 0

R:

\
..
BACKUP
HELPINDX
MACROS
SAMPLES
SMARTAUX
TABLES
TEMP

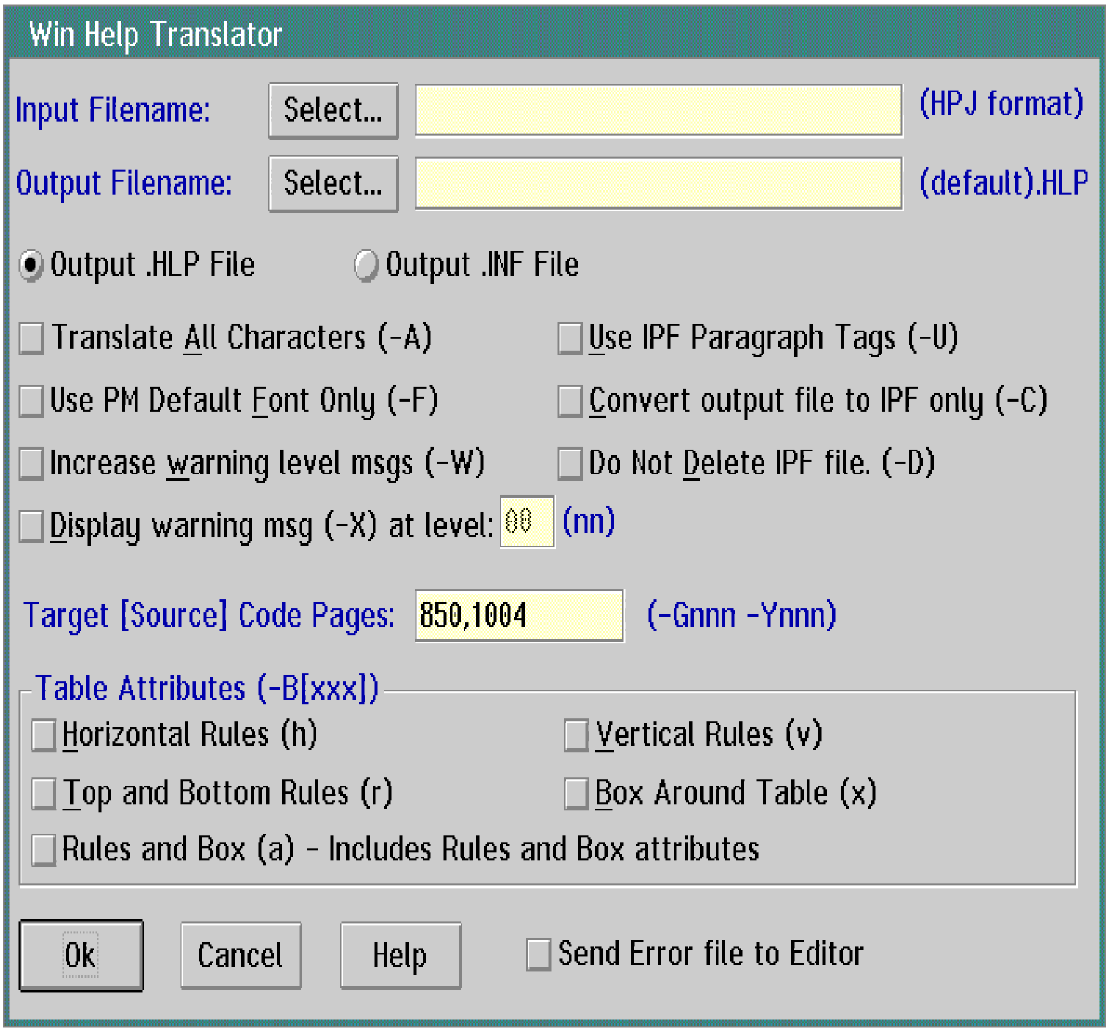
*.HPJ

3. SMART displays the Win Help Translator dialog. Enter the name, including the path, of the HPJ file in the Input Filename entry field.

Choose the rest of the options as they apply. Generally, you will need to use most of the options. These are some of the options you might choose:

- Check the Send error file to editor check box so that SMART logs the status information.
- If the output will be used by an application, choose Output .HLP file.
- If the output will be independent, like a users' reference, choose Output .INF file.
- To view the output, create an INF file. Use the OS/2 VIEW command to view the INF file.
- Unless you choose the Convert to IPF Only option, SMART will produce the IPF file and automatically send it to the IPF compiler. In that case, SMART will produce an INF or HLP file (depending on which option you choose). If you choose the Convert to IPF Only option, SMART produces the IPF file, but does not compile it.

When you choose all the options you need, click on the **Ok** push button.



Input Filename: (HPJ format)

Output Filename: (default).HLP

● Output .HLP File ○ Output .INF File

□ Translate All Characters (-A) □ Use IPF Paragraph Tags (-U)

□ Use PM Default Font Only (-F) □ Convert output file to IPF only (-C)

□ Increase warning level msgs (-W) □ Do Not Delete IPF file. (-D)

□ Display warning msg (-X) at level: 08 (nn)

Target [Source] Code Pages: 850,1004 (-Gnnn -Ynnn)

Table Attributes (-B[xxx])

□ Horizontal Rules (h) □ Vertical Rules (v)

□ Top and Bottom Rules (r) □ Box Around Table (x)

□ Rules and Box (a) - Includes Rules and Box attributes

□ Send Error file to Editor

4. SMART does not display a status on the conversion process for a HELP document. If you choose the Send Error File to Editor option, then the process will be complete when SMART displays the error file. Otherwise, watch the OS/2 window in the background to ensure that the Help conversion process is complete.

5. Output from the Help conversion will depend on which options you choose. If you did not enter anything in the Output Filename entry field, SMART writes the file will to the same directory as the input file.

   Note: The online help explains the error messages. To view the error message explanations, choose HELP from the Win Help Translator dialog panel. Then choose Win Help Translator Topics. After that, choose Win Help Translator Messages.

   Please note that you must have the IPF compiler (IPFC.EXE) for SMART Win Help translation. Information Presentation Facility (IPF) is available from the IBM Developer's Toolkit.

Windows and Open32 Differences

The following are the differences between Windows 32-bit and OS/2 Warp functions' behaviors. These differences were discovered by testing Windows NT and OS/2 Warp functions. References to *Windows* and *Windows 32-bit* in this section apply to the Windows NT operating system.

Accelerator Tables

The following sections describe the differences between Windows and OS/2 accelerator tables.

System Accelerator Table

In Windows, the system accelerator table is processed inside the GetMessage function before an application calls TranslateAccelerator for its own accelerator table. Consequently, in OS/2, all keystrokes handled by the OS/2 system accelerator table will be filtered before returning from GetMessage. Most notably, an OS/2 application will never see the F1 key, which it would see under Windows.

Keyboard Shift States

In Windows, the shift state of a key is ignored unless explicitly specified. For example:

```
ID_MYACCELTABLE ACCELERATORS
  BEGIN
    "a", ID_LOWERA, ASCII
    "B", ID_UPPERB, ASCII SHIFT
  END
```

This accelerator table generates a WM_COMMAND of ID_LOWERA when the user clicks on the "a" key or "Shift+a" keys (with Caps Lock ON). However, a WM_COMMAND of ID_UPPERB will be generated only when you click on the "Shift+b" keys (with Caps Lock OFF).

In OS/2, the shift state of a key must be specified explicitly: The equivalent OS/2 accelerator table would look like:

```
ACCELTABLE ID_MYACCELTABLE
  BEGIN
    "a", ID_LOWERA, CHAR
    "a", ID_LOWERA, CHAR SHIFT /* Additional entry to allow SHIFT+a */
    "B", ID_UPPERB, CHAR SHIFT
  END
```

For Open32, the OS/2 model is followed; accelerator table entries that do not specify SHIFT will not generate WM_COMMAND messages when the user clicks on the Shift key.

Class Names

Windows class names are case insensitive. OS/2 class names are case sensitive. Open32 converts class names to uppercase before registering them with OS/2. In general, Open32 that deal with class names work as expected, with respect to Windows; for example, GetClassInfo returns the class name exactly as it was originally registered. One important difference occurs with resource files, however. Open32 does not currently support Windows resources; Open32 resources are really OS/2 resources. If an Open32 resource contains a user-defined class, the class name must be converted to uppercase because that it how the class is registered internally with OS/2.

Class Styles

The following class styles are not supported by Open32:

CS_CLASSDC
CS_PARENTDC

Clipboard Viewer Chain

In implementations of Win32 other than the OS/2 version, applications are responsible for keeping track of the viewer chain and behaving well. In OS/2, OS/2 keeps track of the viewer chain, meaning that an application will always think that there is no viewer following it in the chain, and applications will never receive the WM_CHANGECBCHAIN message.

Combination Boxes

In Windows, a combination box consists of an edit control, which is a child of the combination box control and a list box control, which is a child of the Desktop window. When the list box is dropped, the list box becomes a child of the combination box and reverts to being a child of the Desktop when it is closed.

In Open32, a combination box consists of an edit control, which is a child of the combination box control and a list box control which is a child of the object window. When the list box is dropped, the list box becomes a child of the combination box and reverts to being a child of the root object window when it is closed.

You will need to slightly modify application code to enumerate and find a combination list box control in Windows. For the starting parameter in your enumeration routines, use the following for each platform:

Windows	Desktop window (GetDesktopWindow)
Open32	Object window (HWND_OBJECT)

When creating an Open32 window that will have combination box children, do not use the CLIPCHILDREN flag.

Common Open and Save File Dialog

In OS/2 there is a default filter of All Files that cannot be removed from the Type of File combination box. In Windows, there is no default filter. If a Win32 developer adds one or more filters (including, perhaps, an All Files filter), the Type of File combination box will still have All Files as a choice.

But, if a developer does add one or more filters, there is no way to have the OS/2 All Files filter come up as the default selection. Developers should be cautioned not to remove the All Files entry if they want the option of selecting it as their default choice.

The actual filtering mechanism also behaves quite a bit differently in OS/2:

- In Win32, filtering is centralized, and the current filter can be changed in two different ways:

    - By typing a filter in the File Name edit box (using wild card characters)
    - By selecting a filter from the List Files Of Type combination box

- In OS/2, the filtering is based on an intersection (or union if the proper flag is set) of the contents of the File Name field and the filter currently selected in the Type of File combination box. This might seem awkward to Windows programmers because an intersection of a file name of *.txt and a filter of *.hlp will not display anything (even if txt and hlp files exist) and a file name of *.* and a filter of *.hlp will display only hlp files.

- In Win32, if a filter is selected from the List Files of Type list and the File Name field currently contains wild card characters, then the File Name field will be updated with the selected filter string.

    For example, if File Name contains *.txt or shell.?pp and the user selects List Files of Type filter of All Files (*.*), then File Name will be set to *.*.

Coordinate Spaces

The maximum coordinates allowed in Win32 are not the same as those allowed in OS/2. These maximums are:

World/Page	-2[31] to 2[31]-1	-2[27] to 2[27]-1
Device	-2[27] to 2[27]-1	-2[15] to 2[15]-1

Passing extreme values to the Open32 graphics display interface (GDI) function, even though the values might fall within the above ranges, is not guaranteed to work. There are many factors that can influence whether or not a particular graphics programming interface (GPI) function with a given set of extreme values will provide the expected behavior or not. It is nearly impossible to establish guaranteed maximum coordinates on a per function basis. Because of this, programming tricks that involve passing maximum values to GDI functions is discouraged and not guaranteed to work. An example of such a trick would be to call FillRect with the maximum allowed coordinates to clear a window. Such a call might not work and should be avoided.

Device Context (DC) for Display Device Driver

Device context (DC) for display device drivers cannot be deleted through the DeleteDC function. A display DC gets deleted automatically when its associated window is deleted. If DeleteDC is called to delete a display DC, a success return code (TRUE) is returned, but the DC does not get physically deleted.

DEVMODE Structure and DeviceCapabilities Function

Open32 supports the Win32 DeviceCapabilities function but does not support the old Windows method of loading the printer driver and accessing an internal DeviceCapabilities function. Any application that tries to use DeviceCapabilities or a DEVMODE structure to set or query a printer driver must have the proper Dynamic Job Properties (DJP)-enabled OS/2 print drivers installed.

Dialog Styles

Windows NT does not support system modal dialog boxes. Therefore, if a user specifies DS_SYSMODAL style in the resource file for the dialog box it is ignored. However, with Open32, the DS_SYSMODAL style is converted by the SMART tool to FCF_SYSMODAL and the dialog is implemented as a system modal dialog box. Windows 3.x does support System Modal dialog boxes.

File Pointers

OS/2 file pointers are limited to 32 bits, whereas in Windows NT, some functions support 64-bit file pointers.

Functions

The following Open32 functions behave differently than their Windows 32-bit counterparts.

AdjustWindowRect

The OS/2 version of AdjustWindowRect does not emulate the Windows 32-bit version. The OS/2 version calculates the adjustment required to accommodate the window that CreateWindow places on the screen, as long as the caller passes the same rectangle structure pointer and style parameters to AdjustWindowRect and CreateWindow.

Note: In Windows, if you call AdjustWindowRect with a NULL style parameter (to indicate the Overlapped style), Windows does not adjust the rectangle. Yet, if CreateWindow is called with the same NULL style parameter, a caption is added. If the caller tries to use the rectangle returned by AdjustWindowRect to size a window of this style, the sizing is off when the window is created. Many other style parameters are treated with this kind of inconsistency in Windows. In terms of window sizing, Windows is consistent with itself only where dialogs and overlapped windows are concerned. These are the only kinds of windows that really have client areas in Windows.

AdjustWindowRectEx

The OS/2 version of AdjustWindowRectEx does not emulate the Windows 32-bit version. Instead, the OS/2 function calculates the amount of adjustment necessary to accommodate the window that CreateWindow places on the screen, so long as the caller passes the same rectangle structure pointer and style parameters to AdjustWindowRectEx and CreateWindow.

Note: In Windows, if you call AdjustWindowRectEx with a NULL style parameter (to indicate the Overlapped style), Windows does not adjust the rectangle. Yet, if CreateWindow is called with the same NULL style parameter, a caption is added. If the caller tries to use the rectangle returned by AdjustWindowRectEx to size a window of this style, the sizing is off when the window is created. Many other style parameters are treated with this kind of inconsistency in Windows. In terms of window sizing, Windows is consistent with itself only where dialogs and overlapped windows are concerned. These are the only kinds of windows that really have client areas in Windows.

BitBlt

Logical color palettes cannot contain more entries than the color-depth of the video driver (for example, on a 256-color driver, you cannot call the BitBlt function with a color palette of more than 256).

The XGA video driver (and all based on it) cannot handle being passed a palette larger than 256 on a machine that has a color-depth of 256. It is possible to do this, but the video driver traps.

Chord

OS/2 allows these maximum coordinates:

- -134217728 to 134217727 (world and page spaces)
- -32768 to 32767 (device spaces)

If you use INT_MAX (2147483647) or INT_MIN (-2147483646) for maximum or minimum coordinates, Chord does not function properly.

CloseClipboard

In OS/2, GlobalFree is not needed to release the memory for CloseClipboard. If GlobalFree is called after the CloseClipboard call, CloseClipboard fails.

CloseEnhMetaFile

Note: This API underwent a limited test cycle. Its quality was deemed acceptable for the applications for which it was developed.

CloseFigure

CloseFigure returns FALSE if no path exists.

CopyEnhMetaFile

Note: This API underwent a limited test cycle. Its quality was deemed acceptable for the applications for which it was developed.

CreateAcceleratorTable

OS/2 does not support FNOINVERT_W. For character VKeys, OS/2 generates two accelerator entries: one uppercase; one lowercase.

CreateBitmap

In OS/2, the bitmap width and height are bounded by screen resolution. For example, for a screen resolution of 1024 x 768, the bitmap:

- Width must be between 0 and 1024
- Height must be between 0 and 768

The plane count usually does not work if greater than 1 because of the restriction of the physical device. OS/2 supports a bits-per-pel count of:

- 1
- 4
- 8
- 24

Any other value fails.

Note: Testing of the Windows 32-bit CreateBitMap function shows that if CreateBitmap is called to get a handle of a one-plane device-dependent bitmap, and the third parameter (cPlanes) is greater than one, a handle is returned as if CreateBimap was successful. However, a subsequent call to BitBlt or PatBlt reveals that this handle is not valid, because even though no error is returned, no bitmap appears. The OS/2 implementation of CreateBitmap fails in this case (that is, if the plane count is greater than one). In other words, the return value from our implementation of CreateBitmap is NULL. This is because the underlying call to GpiCreateBitmap returns an error in this case.

CreateBitmapIndirect

In OS/2, the bitmap width and height are bounded by screen resolution. For example, for a screen resolution of 1024 x 768, the bitmap:

- Width must be between 0 and 1024
- Height must be between 0 and 768

The plane count usually does not work if greater than 1 because of the restriction of the physical device. OS/2 supports a bits per pel count of:

- 1
- 4
- 8
- 24

Any other value fails.

CreateCaret

OS/2 does not support logical unit specification for the size of the caret; the units are always assumed to be in pels. In Windows, the sizes are in logical units.

CreateCompatibleDC

In Windows, it is possible to create a monochrome memory device context (DC) and then BitBlt it to the screen. In OS/2, if you try to do this, the drawing appears black.

In OS/2, do not draw to a monochrome memory DC. In addition, VGA and XGA drivers behave differently when you try to do this in OS/2.

In OS/2, you should instead draw to a color DC and then BitBlt into a monochrome DC.

CreateCursor

As in Windows, the width and height parameters to this function must specify a width and height that are supported by the current display driver. Because OS/2 supports only bitmaps padded to 32 bits (where Windows 32-bit bitmaps are padded to 16 bits), the OS/2 implementation of this function might be more restrictive than the Windows 32-bit implementation depending on the driver.

CreateDialogIndirect

In OS/2, CreateDialogIndirect supports OS/2 dialog templates only; these functions do not accept Windows dialog templates.

CreateDialogIndirectParam

In OS/2, CreateDialogIndirectParam supports OS/2 dialog templates only; these functions do not accept Windows dialog templates.

CreateEllipticRgn

OS/2 supports a range of values for the region coordinates of -32767 to 32766; this is a GPI restriction.

CreateEllipticRgnIndirect

OS/2 supports a range of values for the region coordinates of -32767 to 32766; this is a GPI restriction.

CreateEnhMetaFile

Note: This API underwent a limited test cycle. Its quality was deemed acceptable for the applications for which it was developed.

CreateEvent

OS/2 allows backslashes in the object name; Windows prohibits them.

CreateFile

In Windows, CreateFile assumes that application designers are careful not to overwrite or delete files; CreateFile allows write and delete operations to be performed regardless of file attribute settings and regardless of the manner in which the file is accessed. This behavior might be somehow in keeping with the security implementation of Windows, but it differs from current industry standards on most other platforms.

In Windows NT, there is a privilege level that allows directories and files to be deleted even if it has the read-only file attribute set. This privilege level, called *full control privilege* , applies to the NTFS file system. Judging by the behavior of the CreateFile, WriteFile, and DeleteFile functions on Windows NT, it appears that the full control privilege level is implemented in Windows NT as the default privilege level for FAT and other file systems as well. In Windows NT, if the CREATE_ALWAYS parameter is used, CreateFile deletes any file of the same name that currently exists, regardless of its attributes, and regardless of the access mode set by fdwAccess. The OS/2 version of CreateFile fails if CREATE_ALWAYS is used and the access mode is set to read-only.

CreateFont

The font matching capabilities of Windows NT differ from the capabilities in OS/2. Specifically, some of the parameters that can be passed to the Windows NT version of CreateFont cannot be passed to the OS/2 version of CreateFont:

- nCharSet
- nOrientation
- outPrecis - only OUT_OUTLINE_PRECIS and OUT_STROKE_PRECIS bits are valid
- clipPrecis
- outQual
- nEscapement

- fdwPitchAndFamily - ignore pitch (PROPORTIONAL/FIXED), but honor family

The OS/2 version of CreateFont treats OUT_OUTLINE_PRECIS and OUT_STROKE_PRECIS identically. Although either can be passed to the OS/2 version of CreateFont, only OUT_STROKE_PRECIS is returned by GetObject.

In Windows NT, height represents world coordinates as follows:

0 A default height is to be used by the searching algorithm.

+n The searching algorithm should look for a font that has the specified character cell height (Em height).

-n The searching algorithm should look for a font that has the specified character height (lMaxBaselineExt). The following formula is used:

```
Em height = lMaxBaselinesExt + lInternalLeading
```

In OS/2, the GPI font matching algorithm does not provide a means to search based on the Em height (cell height); it only considers the lMaxBaselinesExt. So, from a Windows 32-bit application perspective, the OS/2 version of CreateFont acts as if it is passed a negative height.

CreateHatchBrush

Although OS/2 tries to provide the closest possible match for each hatch brush in Windows NT, some of the OS/2 hatch patterns do not precisely match the Windows NT hatch brush patterns. The patterns for the OS/2 hatch brushes are from the Base Pattern Symbol set shown in *OS/2 Warp Graphics Programming Interface Programming Guide* .

CreateIconFromResource

The OS/2 version of CreateIconFromResource loads an OS/2 Icon resource.

CreateMutex

OS/2 allows backslashes in the object name; Windows prohibits them.

CreatePen

When a line is drawn, the number of colors that can be displayed at once for that line is:

- Two for Windows NT
- One for OS/2

This difference in drawn colors applies only to lines not to filled regions. If a brush object is selected and visible foreground and background colors are chosen, a two-color pattern is drawn when an area is filled using the brush.

The OS/2 version of CreatePolygonRgn matches the Windows NT behavior for brush objects. However, if a pen object is selected, the OS/2 version draws only one color, no matter what pen style or background mix are chosen. Any gaps in the pattern, as determined by the pen style, will be the color of the region on which the line is drawn. This behavior differs from the Windows NT behavior due to an OS/2 GPI restriction.

CreatePolygonRgn

OS/2 supports a range of values for the region coordinates of -32767 to 32766; this is a GPI restriction. A polygon, as used in Open32 and OS/2, can must have greater th points. A polygon of 2 points is a line.

CreatePolyPolygonRgn

OS/2 supports a range of values for the region coordinates of -32767 to 32766; this is a GPI restriction. A polygon, as used in Open32, must have more than two points. A polygon of two points is a line.

CreateProcess

The OS/2 version of CreateProcess supports these parameters differently than the Windows NT version:

Security attributes	Not supported
Inherit flag	Not supported
STARTUPINFO	lpTitle, dwX, dwY, dwSizeX, dwSizeY, wShowWindow, PgmControl

----

CreateRectRgn

OS/2 supports a range of values for the region coordinates of -32767 to 32766; this is a GPI restriction.

CreateRectRgnIndirect

OS/2 supports a range of values for the region coordinates of -32767 to 32766; this is a GPI restriction.

CreateRoundRectRgn

OS/2 supports a range of values for the region coordinates of -32767 to 32766; this is a GPI restriction.

CreateSemaphore

OS/2 allows backslashes in the object name; Windows prohibits them.

CreateWindow

The OS/2 version of CreateWindow differs from the Windows NT version where negative client areas are concerned. In Windows NT, if the client area has negative height or width, the window is drawn as though that height or width is zero. In OS/2, the absolute value of the negative space is subtracted from the surrounding border, title bar, and so on.

For example, assume that a window is to be created, and according to the parameters passed into CreateWindow:

- The bottom of the window's client area is located at 100 (device coordinates)
- The top of the caption is at 72 (device coordinates)

This would place the bottom of the client area at 95, depending on the height of the caption in device coordinates. In other words, the bottom of the client area is five pixels above the top.

With the same parameters, in Windows NT, a complete caption nonetheless appears on-screen when the window is created. In contrast, in OS/2, the bottom of the caption is clipped off.

DdeUnaccessData

This is a Windows NT-specific function. DdeUnaccessData unlocks global memory previously locked by a call to DdeAccessData. Because OS/2 memory management architecture does not impose such requirements, this function is not used in OS/2.

DeleteAtom

The documented behavior for this function in Windows NT is to return the atom if the atom to be deleted does not exist; however, the Windows NT version of this function returns 0 in all cases. The OS/2 version of DeleteAtom returns the atom if the atom to be deleted does not exist.

DeleteEnhMetaFile

Note: This API underwent a limited test cycle. Its quality was deemed acceptable for the applications for which it was developed.

DialogBoxIndirect

In OS/2, DialogBoxIndirect and DialogBoxIndirectParam support the passing of OS/2 templates.

DllEntryPoint

The OS/2 version of DllEntryPoint does not support:

- The Windows 16-bit entry point LibMain
- In Windows NT, DllEntryPoint can be called on thread creation and destruction. OS/2 does not support this.
- In Windows NT, DllEntryPoint has a parameter that distinguishes between dynamic and static loading. OS/2 does not support this flag.

DrawText

The OS/2 and Windows NT versions of this function differ as follows:

- Windows NT uses the last mnemonic on a line (the other mnemonic characters are treated as normal characters); the OS/2 implementation uses the first mnemonic as the mnemonic.
- Windows NT does not allow DT_MULTILINE to be mixed with DT_BOTTOM or DT_VCENTER. The OS/2 implementation does.
- OS/2 does not support the Windows NT undocumented DT_INTERNAL flag.
- The OS/2 version has the following new flags:

  DT_AMPERSAND
  Causes the ampersand to be recognized as the mnemonic character instead of the tilde.

  DT_MULTILINE
  Enables WinDrawText to output two or more lines of text in one operation. DT_MULTILINE is mutually exclusive with DT_SINGLELINE and DT_ERASERECT; if used in combination with these, the result is undefined.

  DT_NOCLIP
  Text is not clipped to the rectangle passed in. Can improve performance and alter the text output.

  DT_OPAQUE
  Causes the text background rectangle to be filled with the background color.

  DT_SINGLELINE
  Causes linefeeds and carriage returns to be treated like normal characters.

  DT_VERTICALEXTENT
  If DT_LINEEXTENT is set, WinDrawText and DrawText returns the height of text drawn (or potentially drawn) instead of the number of characters printed.

- In OS/2, the DT_EXTERNALLEADING flag might affect how text is drawn if the font used has an non-zero external leading value, especially if multiple lines of text are output (DT_MULTILINE). Previously, this flag affected only the extent of the rectangle returned when DT_QUERYEXTENT was set.

DuplicateHandle

File handles cannot be duplicated from or to processes that are not children of the calling process or the process itself, because this is not supported by OS/2 file systems.

Ellipse

OS/2 allows the following maximum coordinates:

- -134217728 to 134217727 (world and page spaces)

- -32768 to 32767 (device spaces)

If INT_MAX (2147483647) or INT_MIN (-2147483646) are used for maximum or minimum coordinates, the function does not function properly.

EnableScrollBar

In OS/2, this function cannot enable or disable individual arrow buttons: either both buttons are enabled or both are disabled.

EndPath

EndPath returns FALSE if no path exists or the path has been destroyed with AbortPath.

EnhMetaFileProc

Note: This API underwent a limited test cycle. Its quality was deemed acceptable for the applications for which it was developed.

EnumEnhMetaFile

Note: This API underwent a limited test cycle. Its quality was deemed acceptable for the applications for which it was developed.

EnumMetaFile

In OS/2, the metafile handle retrieved from the clipboard or loaded from an OS/2 metafile does not work because it only contains OS/2 metafile handle and has no information about the actual metafile records.

Escape

In OS/2, the following escapes are supported:

- ABORTDOC
- ENDOC
- GETPHYSPAGESIZE
- GETPRINTINGOFFSET
- GETSCALINGFACTOR
- NEWFRAME
- STARTDOC

ExcludeClipRect

OS/2 supports a range of values for the rectangle coordinates of -32767 to 32766; this is a GPI restriction.

ExtCreateRegion

OS/2 supports a range of values for the region coordinates of -32767 to 32766; this is a GPI restriction.

FillRect

In Windows NT, FillRect accepts invalid rectangle values and does not fail. The Windows 16-bit of this function did not have this behavior.

In OS/2, this function returns an error if parameters are invalid.

FindResource

OS/2 does not support resource types:

- RT_GROUPCURSOR
- RT_GROUPICON
- RT_VERSION

To load resource with RT_STRING type, the resource id has to be computed before calling LoadResource due to the way OS/2 stores the string table data. Use the formula:

```
 (string id / 16) + 1
```

Similarly, for RT_MESSAGETABLE, compute resource id as:

```
 (message id / 16) + 1
```

FrameRgn

OS/2 supports a range of values for the region coordinates of -32767 to 32766; this is a GPI restriction.

FreeProcInstance

In OS/2, this is a NULL macro.

GdiComment

Note: This API underwent a limited test cycle. Its quality was deemed acceptable for the applications for which it was developed.

GetAsyncKeyState

If GetAsyncKeyState is issued twice in succession while the key is down, the second GetAsyncKeyState reports the key as being up.

Virtual keys for regular characters like VK_A, VK_B, and so on are not supported -- just as they are not supported for GetKeyState.

For mouse Vkeys like VK_LBUTTON, the synchronous state, rather than the asynchronous state, is provided becasue PM does not provide any way to get the asynchronous state of the mouse buttons.

GetClassInfo

In OS/2, this function fails if called against a window not created through Open32, because it returns class information specific to Open32 window classes.

GetClassLong

In OS/2, this function fails if called against a window not created through Open32, because it returns class information specific to Open32 window classes.

GetClassWord

In OS/2, this function fails if called against a window not created through Open32, because it returns class information specific to Open32 window classes.

GetClientRect

The OS/2 version of GetClientRect differs from the Windows NT version where negative client areas are concerned. In Windows NT, if the client area has negative height or width, the window is drawn as though that height or width is zero. In OS/2, the absolute value of the negative space is subtracted from the surrounding border, title bar, and so on.

For example, assume that a window is to be created, and according to the parameters passed into CreateWindow:

- The bottom of the window's client area is located at 100 (device coordinates)
- The top of the caption is at 72 (device coordinates)

This would place the bottom of the client area at 95, depending on the height of the caption in device coordinates. In other words, the bottom of the client area is five pixels above the top.

With the same parameters, in Windows NT, a complete caption nonetheless appears on-screen when the window is created. In contrast, in OS/2, the bottom of the caption is clipped off.

GetClipboardViewer

In OS/2, GetClipboardViewer returns a valid window handle that can be used to send messages to all viewers in the system (including one registered through WinSetClipbrdViewer). Clipboard messages and messages with a value greater than WM_USER sent to this window are sent to all viewers known to the system. However, the window returned by GetClipboardViewer will not be a viewer window set by an application.

GetEnhMetaFile

Note: This API underwent a limited test cycle. Its quality was deemed acceptable for the applications for which it was developed.

GetEnhMetaFileBits

Note: This API underwent a limited test cycle. Its quality was deemed acceptable for the applications for which it was developed.

GetEnhMetaFileDescription

Note: This API underwent a limited test cycle. Its quality was deemed acceptable for the applications for which it was developed.

GetEnhMetaFileHeader

Note: This API underwent a limited test cycle. Its quality was deemed acceptable for the applications for which it was developed.

GetEnhMetaFilePaletteEntries

Note: This API underwent a limited test cycle. Its quality was deemed acceptable for the applications for which it was developed.

GetFileTime

The GetFileTime function is related to the OS/2 DosQueryFileInfo function. GetFileTime returns whatever the DosQueryFileInfo function returns. This means that the function might return:

- 0 creation time
- 0 last access time

GetKeyboardState

Virtual keys for regular characters like VK_A, VK_B, and so on are not supported -- just as they are not supported for GetKeyState.

GetLocaleInfo

LOCALE_ILDATE is not supported on OS/2.

LOCALE_ITIMEMARKPOSN always returns 0.

LOCALE_INEGNUMBER always returns 1.

LOCALE_SGROUPING and LOCALE_SMONGROUPING always return the grouping size followed by ";0" to match what Windows NT provides. For example, for the US, the grouping is 3 so the value of LOCALE_SGROUPING and LOCALE_SMONGROUPING is "3;0".

LOCALE_STIMEFORMAT returns "h:mm:ss tt" on Windows NT, and "h:m:s" on OS/2.

LOCALE_SLONGDATE returns "dddd, mmmm dd, yyyy" on Windows NT, and "mm/dd/yy" on OS/2.

LOCALE_SSHORTDATE returns "M/d/yy" on Windows NT, and "M-d-yy" on OS/2.

GetMetaFileBitsEx

There are two different kinds of Open32 metafiles:

1. A true Open32 metafile that contains a stream of GDI commands.

2. A metafile that has been converted into an Open32 metafile. A metafile of this type is created by obtaining a metafile handle through WinTranslateGraphicsObject. These metafiles are wrappers for an OS/2 metafile.

You can use GetMetaFileBitsEx with the first metafile type only. This function does not work with the second type of metafile: a converted metafile.

Note: This API underwent a limited test cycle. Its quality was deemed acceptable for the applications for which it was developed.

GetTimeZoneInformation

When you use GetTimeZoneInformation, the time zone setting defaults to Greenwich Mean Time (GMT) with UTC bias set to 0. You can use SetTimeZoneInformation to change the time zone setting.

GetVersion

This function returns Windows NT Version 3.51.

GetVersionEx

This function returns Windows NT Version 3.51.

GetWinMetaFileBits

Note: This API underwent a limited test cycle. Its quality was deemed acceptable for the applications for which it was developed.

IsIconic

In OS/2, IsIconic works only with frame windows.

LineTo

In OS/2, the coordinates for LineTo round differently than they do in Windows. In OS/2, a line drawn from (1,10) to (10.5,10) results in a line drawn to (11,10), not (10,10) as would happen under Windows NT. This can result in stray pixels or unexpected overlap of lines.

LoadLibrary

In OS/2, when LoadLibrary is called to load a device driver without specifying the complete path to the file, OS/2 might not find the driver. In Windows, device drivers are typically placed in the Windows system directory. In OS/2, they are placed in the os2\dll subdirectory. This subdirectory is typically not in any path so LoadLibrary does not find the drivers unless the complete path is specified.

----

lstrcmpW

OS/2 sorts by character value; Win32 sorts using "dictionary" order:

1. Non-alphanumeric
2. Numbers
3. Letters

MessageBox

In OS/2, when using MessageBox, the width of the message box is adjusted to fit the title and the buttons, but no adjustment is made for the message itself. This might cause portions of the message text to be forced to a new line.

MultiByteToWideChar

A single ASCII character can map to more than one Unicode character. For example, 0xE6 in ASCII maps to Unicode 0x3BC and Unicode 0xB5. In some instances, OS/2 returns a different character value.

PlayEnhMetaFile

Note: This API underwent a limited test cycle. Its quality was deemed acceptable for the applications for which it was developed.

PlayEnhMetaFileRecord

Note: This API underwent a limited test cycle. Its quality was deemed acceptable for the applications for which it was developed.

RealizePalette

The OS/2 version of RealizePalette is valid only with a screen DC.

RedrawWindow

In OS/2, RedrawWindow does not support redraw flags:

- RDW_FRAME
- RDW_INVALIDATE
- RDW_NOINTERNALPAINT

Registry Functions

OS/2 supports the following predefined key handles:

- HKEY_CLASSES_ROOT

- HKEY_CURRENT_USER
- HKEY_LOCAL_MACHINE
- HKEY_USERS
- REGH_INIMAPPING
- REGH_SYSINFO
- REGH_WINOS2INI

For HKEY_LOCAL_MACHINE, OS/2 accepts and converts only the following list of subkeys:

- All new keys added under HKEY_LOCAL_MACHINE\SOFTWARE\MICROSOFT\WINDOWS and NT\CURRENTVERSION\INIFILEMAPPING is added to REGH_INIMAPPING.
- All references to HKEY_LOCAL_MACHINE\SOFTWARE are mapped to REGH_WINOS2INI (application's data).

OS/2 does not support REG_LINK and REG_RESOURCE_LIST value types.

The registry supplies mappings for the following subkeys:

`HKEY_CLASSES_ROOT\ ... (alias to HLM\Software\Classes)`	`REGH_SYSINFO\Classes\ ...`
`HKEY_CURRENT_USER\Software\ ...`	`REGH_SYSINFO\User\Current\ ...`
`HKEY_LOCAL_MACHINE\Software\ ...`	`REGH_WINOS2INI\  ...`
`HKEY_LOCAL_MACHINE\Software\Classes\ ...`	`REGH_SYSINFO\Classes\ ...`
`HKEY_LOCAL_MACHINE\System\CurrentControlSet\ h. ...`	`REGH_SYSINFO\Control\ ...`
`HKEY_USERS\.Default\ ...`	`REGH_SYSINFO\User\Default\ ...`

Notes:

- HKEY_CLASSES_ROOT is totally unrestricted with regard to the first subkey and it is an alias to the information stored in the registry key *HKEY_LOCAL_MACHINE\Software\Classes* .
- HKEY_CURRENT_USER must have *Software* as the first subkey.
- HKEY_LOCAL_MACHINE must have either *Software* or *System\CurrentControlSet* as the first subkey.
- HKEY_USERS must have *.Default* as the first subkey (note the period).

Resource Functions

The following functions are not source-compatible with Windows:

- CreateBitmapIndirect
- CreateDialogIndirect
- CreateDialogIndirectParam
- CreateIconIndirect
- DialogBoxIndirect
- DialogBoxIndirectParam
- LoadMenuIndirect
- LoadResource

SetClipboardViewer

In Windows NT, applications keep track of the viewer chain through this function and ChangeClipboardChain. In OS/2, OS/2 keeps track of the viewer chain.

In OS/2, only one viewer is allowed and OS/2 performs no arbitration when an application sets itself as the viewer; OS/2 just ends the old viewer with no warning. Each application thinks that it is the only viewer in the system.

In OS/2, SetClipboardViewer always returns 0. OS/2 does not tell applications about other viewers. In OS/2, it is not the application's responsibility to pass clipboard messages on to the next viewer in the chain. In Windows NT, 0 indicates there is no next viewer.

In OS/2, applications are never sent WM_CHANGECBCHAIN; this message is meaningless in OS/2. In Windows NT, the viewer that is leaving the chain does not get this message; in OS/2, the other viewers do not know about each other so this message is meaningless to them. In Windows NT, they would just pass it on.

SetEnhMetaFileBits

Note: This API underwent a limited test cycle. Its quality was deemed acceptable for the applications for which it was developed.

SetKeyboardState

Virtual keys for regular characters like VK_A, VK_B, and so on are not supported -- just as they are not supported for GetKeyState.

SetLocaleInfo

LOCALE_ILDATE is not supported on OS/2.

LOCALE_ITIMEMARKPOSN always returns 0.

LOCALE_INEGNUMBER always returns 1.

LOCALE_SGROUPING and LOCALE_SMONGROUPING always return the grouping size followed by ";0" to match what Windows NT provides. For example, for the US, the grouping is 3 so the value of LOCALE_SGROUPING and LOCALE_SMONGROUPING is "3;0".

LOCALE_STIMEFORMAT returns "h:mm:ss tt" on Windows NT, and "h:m:s" on OS/2.

LOCALE_SLONGDATE returns "dddd, mmmm dd, yyyy" on Windows NT, and "mm/dd/yy" on OS/2.

LOCALE_SSHORTDATE returns "M/d/yy" on Windows NT, and "M-d-yy" on OS/2.

SetTimeZoneInformation

When you use GetTimeZoneInformation, the time zone setting defaults to Greenwich Mean Time (GMT) with UTC bias set to 0. You can use SetTimeZoneInformation to change the time zone setting.

SetWinMetaFileBits

Note: This API underwent a limited test cycle. Its quality was deemed acceptable for the applications for which it was developed.

ShellExecute

Only the "open" command is supported (no "print"). Opening of folders is not supported.

The ShowCmd flags are only available if the application being launched has FCF_SHELLPOSITION specified.

The SW_HIDE show command is not supported.

On successful execution, the HRPROCESS returned is always -1.

StrokePath

In OS/2, StrokePath returns TRUE only if a path exists in the device context; otherwise, StrokePath returns FALSE.

StrokeAndFillPath

In OS/2, StrokeAndFillPath returns TRUE only if a path exists and is closed in the device context; otherwise StrokeAndFillPath returns FALSE.

TrackPopUpMenu

In Open32, TrackPopUpMenu ignores the last parameter, which specifies a rectangular box that is used specify an area that the user can click in to keep the track pop menu on the screen. For this parameter, Open32 uses the NULL value response, which is to remove the menu any time the user clicks.

WinHelp

The OS/2 WinHelp implementation supports only the following commands:

- HELP_CONTENTS: Displays contents panel and not the contents option specified in .HPJ file.
- HELP_CONTEXT
- HELP_PARTIALKEY: Only a null string input can be accepted, which causes the search dialog box to be displayed.
- HELP_QUIT

As in Windows, the second parameter of WinHelp-lpszHelpFile- accepts the help file name with path included. But, the path must be defined in the current SET HELP path. Otherwise, an error code is returned. In OS/2, all help files are expected to reside in a directory that is defined in *SET HELP* path. Otherwise, the Help Manager is not able to locate them.

There is no secondary window support for lpszHelpFile parameter and no macro support.

WinMain

To be able to use the Windows NT WinMain() function, use the OS/2 main() function located in the following subdirectory:

```
TOOLKIT\BETA\SAMPLES\DAPIE\WINMAIN\ma
in.c
```

You can also use the main function, a DLL, in the following subdirectory:

```
 TOOLKIT\BETA\SAMPLES\DLLMAIN\dllmain.c
```

main.c gets compiled and linked with the module containing WinMain and creates an OS/2 Warp executable. If you do not use the OS/2 Warp main function, you receive a link error stating that there is no starting address for your program.

GDI Return Codes

In Windows NT, GDI functions send messages to the graphics subsystem rather than making direct function calls. To improve performance, functions that do not return a value, other than a return code, send the message asynchronously and immediately return TRUE. The requested action might subsequently fail, but the caller will not be informed of this. In OS/2, the graphics operations are performed during the GDI function call, so OS/2 validates the parameters and returns a useful return code. There are instances when OS/2 returns FALSE and Windows NT returns TRUE.

GPI Mode Parameter

For functions such as SetBkMode and SetPolyFillMode, Windows NT allows the functions to succeed even with incorrect values specified for the mode parameter.

Line Attributes

OS/2 supports line styles only for cosmetic lines. This means that when a line is drawn as a geometric line, the line is always solid, which occurs in the following circumstances:

- The line width is greater than one pel.
- The lines are drawn in a path.
- The functions Pie and Chord, which use paths.

Logical Color Palettes

Logical color palettes cannot contain more entries than the color-depth of the video driver (for example, on a 256-color driver, you cannot call the BitBlt function with a color palette of more than 256).

The XGA video driver (and all based on it) cannot handle being passed a palette larger than 256 on a machine that has a color-depth of 256. It is possible to do this, but the video driver traps.

Menus

Windows supports inactive menu items (disabled but not greyed) as well as OS/2-style disabled menu items. Because OS/2 does not have an equivalent concept, Open32 does not support this functionality.

Messages

The following describes the differences between Open32 and Windows message behavior.

WM_DRAWITEM

The WM_DRAWITEM message for a combination box or a list box are not exactly identical to their Windows counterpart. In Windows, when the item selection is changed using the mouse or keyboard, two WM_DRAWITEM messages are sent for the item deselected and four WM_DRAWITEM messages are sent for the item being selected.

For the deselected item, the messages received are:

A	ODA_FOCUS	ODS_SELECTED
B	ODA_SELECT	0

For the selected item, the messages received are:

C	ODA_FOCUS	ODS_FOCUS
D	ODA_FOCUS	0
E	ODA_SELECT	ODS_SELECTED
F	ODA_FOCUS	ODS_FOCUS | ODS_SELECTED

The order of messages received, assuming the list box had the focus, is:

A
C
D
B
E
F

If the list box did not have the focus and the user clicks in the list box on a new item (such that the list box is also gaining the focus), only messages A, E, and F are sent.

If the list box is just receiving the focus but no new item is being selected, only message F is sent for the item currently selected.

When the list box loses the focus, only message A is sent for the currently selected item.

A special case arises when the user is at the beginning or end of the list and re-selects the selected item using the mouse or keyboard. In this case, only message A is sent.

In the Open32 implementation, one WM_DRAWITEM message is sent for the item deselected and one WM_DRAWITEM message for the item being selected.

For the deselected item, the message received is:

A	ODA_SELECT	0

For the selected item, the message received is:

B	ODA_FOCUS	ODS_FOCUS | ODS_SELECTED

The order of messages received is A, then B. If the list box did not have the focus and the user clicks in the list box on a new item (such that the list box is also gaining the focus), messages A and B are again sent. If the list box is just receiving the focus, only message B is sent. When the list box loses the focus, the following message is sent for the currently selected item:

C	ODA_FOCUS	ODS_SELECTED

For the special case when the user is at the beginning or end of the list and re-selects the selected item using the mouse or keyboard, no message is sent in Open32.

When an item is just to be drawn (in response to painting the list is), the WM_DRAWITEM message received in both implementations is:

A	ODA_DRAWENTIRE	0

In both implementations, the WM_DRAWITEM message received for the combination box edit are:

When the edit control receives focus:

A	ODA_DRAWENTIRE	ODS_COMBOBOXEDIT ODS_FOCUS ODS_SELECTED

When the edit control loses focus:

A	ODA_DRAWENTIRE	ODS_COMBOBOXEDIT

This implementation can cause problems if the drawing of the item's focus relies on only the ODA_FOCUS Action and the ODS_FOCUS state flags. Likewise, problems can occur drawing the item's selection state relies on only the ODA_SELECT Action and ODS_SELECTED State flags. By properly arranging the logic for the WM_DRAWITEM message, an application can be written to work efficiently under both Windows and Open32.

Metafiles

There are two different kinds of Open32 metafiles:

1. A true Open32 metafile that contains a stream of GDI commands.

2. A metafile that has been converted into an Open32 metafile. A metafile of this type is created by obtaining a metafile handle through WinTranslateGraphicsObject. These metafiles are wrappers for an OS/2 metafile.

You can use the GetMetaFileBitsEx function with the first metafile type only. GetMetaFileBitsEx does not work the the second type of metafile: a converted metafile.

Monochrome Memory Device Context (DC)

In Windows, it is possible to create a monochrome memory device context (DC) and then BitBlt it to the screen. In OS/2, if you try to do this, the drawing appears black.

In OS/2, do not draw to a monochrome memory DC. In addition, VGA and XGA drivers behave differently when you try to do this in OS/2.

In OS/2, you should instead draw to a color DC and then BitBlt into a monochrome DC.

Open32 Controls

All Open32 controls, unlike Windows controls, notify their parent when focus changes. Care needs to be taken when processing WM_COMMAND messages from controls so that proper handling based on control type is accomplished. You may receive a WM_COMMAND notification from Open32 that you do not receive from Windows causing unpredictable results. A well-structured program should not encounter problems related to these extra notifications. Open32 applications can also be enhanced to take advantage of these additional notifications to provide OS/2-only additional functionality.

Pattern Brushes

In Windows, pattern brushes can be created with bitmaps of any size. Many of the OS/2 video drivers will only work properly with bitmaps whose widths are divisible by 8 (but there is no guarantee of consistency), and OS/2 VGA supports only pattern bitmaps that are specifically 8*8 (if a larger bitmap is specified, all but the lower left 8*8 goes unused).

Read-Only Files

In Windows NT, there is a privilege level that allows a directory or file to be deleted even if it has the read-only file attribute set. This privilege level is called Full Control privilege. Full Control privilege, like other file system security features in Windows NT, applies to the NT file system (NTFS). Based on the behavior of Windows NT CreateFile and other file-manipulation functions, it appears that the Full Control privilege level in Windows NT is the default privilege level for FAT and other file systems as well. The Windows NT implementation of file access privileges differs from the OS/2 implementation:

FAT	Never Full Control privilege	Always Full Control privilege
HPFS	Never Full Control privilege	Always Full Control privilege
NTFS		Optional Full Control privilege

In OS/2, the OS/2 behavior is preserved instead of emulating the Windows NT behavior with regards to the handling of read-only files. For example, CreateFile can be passed a parameter called CREATE_ALWAYS. If the file to be created has the same name as a file that already exists, the CREATE_ALWAYS parameter causes the existing file to be overwritten. In Windows NT, if the CREATE_ALWAYS parameter is used, CreateFile deletes the existing file, regardless of its attributes, and regardless of the access mode set by fdwAccess (another parameter that tells CreateFile to open the file as writable or read-only). In OS/2, CreateFile fails if CREATE_ALWAYS is used and the access mode is set to read-only. In OS/2, CreateFile also fails if any attempt is made to open an existing file with write access when the file's read-only attribute is set. The OS/2 implementation of other file-manipulation functions, including DeleteFile and WriteFile, similarly respects file attributes.

Programmers using the OS/2 implementation should be aware that a file's attributes must be set appropriately before the file can be deleted or opened with write access. Furthermore, the implementation requires that a file must be opened with write access before data can be written to the file.

Registry

The following general information applies to OS/2 registry. See also Registry Functions for information about the registry functions.

Registry Tree Structure

OS/2 defines the base set of persistent keys in the DOS branch of the name space, hanging off WINOS2 node. The root of these registry keys is called *winreg* . Beneath winreg, the following structure exists:

```
winreg (root)  ---
                  |
                  |-- REGH_WINOS2INI
                  |                  |--- WIN.INI
                  |                                |- WINI_FONTS
                  |                                |- WINI_INTL
                  |                                |- WINI_DEVICES
                  |                                |- WINI_WINDOWS
                  |                  |--- xxxxxxxxx (application ini file name)
                  |
                  |-- REGH_SYSINFO
                  |              |--TIME
                  |                 |- TIME_SETTING
                  |
                  |
                  --- REGH_INIMAPPING (All mappings take place
                                       under REGH_WINOS2INI directory)
                               |----WIN.INI
                                            (contains value name "FONTS"
                                             with value "WIN.INI\WINI_INTL")
                                            (contains value name "DEVICES"
                                             with value "WIN.INI\WINI_DEVICES")
                                            (contains value name "WINDOWS"
                                             with value "WIN.INI\WINI_WINDOWS")
```

winreg is a name space node, not accessible through Windows 32-bit registry functions. In addition to actual key handles, you can use the following as input handles to Windows 32-bit registry functions:

- REGH_WINOS2INI
- REGH_SYSINFO
- REGH_INIMAPPING

They are defined in os2win.h.

The following describes the subdirectories in the tree:

REGH_WINOSINI Contains the supported system INI keys as well as application INI data. All node names are:

- Case-insensitive
- Converted to uppercase

All attribute names are:

- Case-insensitive
- Converted to uppercase, except for value names under WINI_FONTS, WINI_INTL, and WINI_DEVICES

Values for WINI_FONTS, WINI_INTL, and WINI_DEVICES are obtained from OS2.INI and so maintain the case-sensitive quality.

There is only one value contained in WINI_WINDOWS: the default printer device (value name *device*).

REGH_SYSINFO Contains any system information. For now, the only data stored under this branch is time-related information, such as time bias, standard bias, daylight bias, and so on. The information is used by Windows time-related functions.

REGH_INIMAPPING Contains INI file mapping information.

Adding an Application INI File to Registry

Applications can add their own INI file to registry by:

1. Updating REGH_INIMAPPING: Add the name of the application INI file under REGH_INIMAPPING. For example, for application XYZ, create a new node under REGH_INIMAPPING called XYZ.INI. For each section in XYZ.INI, create a value under the XYZ.INI node. The value name is the section name. The value content is the corresponding directory path under the REGH_WINOS2INI node. See step 2.

2. Updating REGH_WINOS2INI: Add the name of the application INI file under REGH_WINOS2INI. For example, for application XYZ, create a new node under REGH_WINOS2INI called XYZ.INI. Add a new node under XYZ.INI. The name of the node is the group name.

After completing these two steps, the application can use Windows 32-bit profile functions to modify the contents of XYZ.INI.

When finished, the registry structure looks like this:

```
winreg (root)  ---
                  |
                  |-- REGH_WINOS2INI --- XYZ.INI
                  |
                                          |--section 1
                                          |
                                          |  (value name=key name 11;
                                          |   value content...)
                                          |   value name=key name 12;
                                          |   value content...)
                                          |
                                          |--section 2
                                          |
                                             (value name=key name 21;
                                              value content...)
                                              value name=key name 22;
                                              value content...)

                  |-- REGH_INIMAPPING
                        |--- XYZ.INI
                        (value name=section1;
                         value content="XYZ.INI\section1")
                        (value name=section2;
                         value content="XYZ.INI\section2")
```

All new keys added under HKEY_LOCAL_MACHINE\SOFTWARE\MICROSOFT\WINDOWS NT\CURRENTVERSION\INIFILEMAPPING are added to REGH_INIMAPPING.

All references to HKEY_LOCAL_MACHINE\SOFTWARE are mapped to REGH_WINOS2INI.

Viewing Registry

The NameExt is an OS/2 command-line utility program that is used to display the contents of nodes in the OS/2 name space. The Open32 registry is part of the OS/2 name space. It resides under the node path DOS/WINOS2/WINREG.

By default, the information generated by this utility program is written to the terminal (STDOUT); however, the user can use the "-f" parameter to write the output to a file. NameExt has the flexibility to provide information to a single node (the default or -1 0), for all nodes at or below the specified node (-1 *), or to limit the depth in the name space. The user has even finer control in being able to select only nodes that satisfy a specific set of search criteria (the "-s" parameter). Finally, the user can:

- Retrieve all information for a node (the default)
- Retrieve only the metadata for a node (the "-m" parameter)
- Include or exclude a list of attributes (the "-i" and "-x" parameters)

Character strings will be output in display format if they contain all printable characters; otherwise they will be output in character-based hex format.

Syntax:

```
>>  NameExt  path                                                           >
                     -s    search_filter

                                    (3)
>                                -m                                         >
           ,                               -f output_file

                        (1)
     -i   attribute_name
           ,

                        (2)
     -x   attribute_name

     -l 0
>                                                                         ><
     -l levels_to_extract
     -l *

SEARCH_FILTER:
       logical_operator

       filter

LOGICAL_OPERATOR:
     AND
     OR
     XOR

FILTER:
     attribute_name   EQ      value_expression
                      NE
                      GT
                      GE
                      LT
                      LE
                           (4)
     EXISTS(attribute_name)
                                   (5)
     RSTR(attribute_name) EQ string
                                   (6)
     LSTR(attribute_name) EQ string
                                     (7)
     MIDSTR(attribute_name) EQ string
     (   filter   )
     NOT(   filter   )

VALUE_EXPRESSION:
     ([system_type                ]value)
                     :user_type
                                        ,

     ([system_type[]                ]  value  )
                       :user_type
     quoted_character_string
         ,

     {  value_expression  }

NOTES:
(1)  This form specifies that the list of attributes that follows should be
     included in the extract.

(2)  This form specifies that the list of attributes that follows should be
     excluded from the extract.

(3)  This form specifies that the extract contains only the metadata and the
     path name.

(4)  This expression is true if the attribute exists and has any value or no
     value.
```

```
(5)  This expression is true if the right-most portion of attribute_name's
     value equals the string.

(6)  This expression is true if the left-most portion of attribute_name's
     value equals the string.

(7)  This expression is true if the string appears anywhere in
     attribute_name's value.
```

Parameters:

path Name space to be displayed

-s Flag used to specify that you want to extract nodes which satisfy a certain set of search criteria.

search_filter The search criteria used to extract nodes.

-i Flag used to specify that you want to provide a list of attributes to be included in the extract.

-x Flag used to specify that you want to provide a list of attributes to be excluded from the extract.

attribute_name The name of the attribute to be extracted.

-m Flag used to specify that you want only the metadata for a node extracted.

-f Sends the results to an output file that you specify.

output_file A user-specified file to which the extracted data is sent.

-l Flag used to specify the number of levels deep (relative to the path) in the name space to extract.

* Flag used to specify that you want all the nodes at or below a specified node.

0 Provides information for the node specified by the path.

levels_to_extract Limits the depth of a name space.

logical_operator Allowable logical operators are AND, OR, and XOR.

value_expression A value given to an expression.

EXISTS(attribute_name)
This expression is true if the attribute exists, and has any or no value.

RSTR(attribute_name) EQ string
This expression is true if the right-most portion of attribute_name's value equals the string.

LSTR(attribute_name) EQ string
This expression is true is the left-most portion of attribute_name's value equals the string.

MIDSTR(attribute_name) EQ string
This expression is true if the string appears anywhere in attribute_name's value.

system_type System-defined attribute types. The system_type can be one of the following:

BOOLEAN
CHAR
UCHAR
UNICHAR
INT8
UINT8
INT16
UNIT16
INT32
UINT32
BYTE
PORT

user_type User-defined attribute types.

value Any value appropriate for the system type.

quoted_character_string
One-dimensional array of characters, enclosed in quotes.

Return Codes:

0	OK. The command ran successfully.
4	Warning. The command probably ran successfully.
8	Error. The command did not run successfully.
12	Internal Error. The command did not run successfully.

There are currently no warnings (return codes of 4) issued by the utilities. A warning return code is included only for completeness. The error will be further identified by messages that are issued by the commands when errors are detected.

Example:

The following example displays the contents of the entire Open32 registry.

```
nameext dos\winos2\winreg -l *
```

Restoring Registry

To restore registry, do the following:

1. End the RSRV session by:

   a. Editing CONFIG.SYS and removing the following line:

      ```
      RUN=...\RSRV.EXE
      ```

      Note: This line will be added back later.

   b. Restarting the system.

2. Delete all *.DAT files in the root directory of the boot drive:
   BOOT.DAT DOS.DAT REGISTRY.DAT

3. Add the following line back in the CONFIG.SYS file:

   ```
   RUN=...\RSRV.EXE
   ```

4. Start up an RSRV session by going to the OS2 directory and typing:

   ```
   start rsrv
   ```

5. Go to the OS2 directory and type:

   ```
   inst_dos dos
   ```

6. Then, type:

   ```
   reginit
   ```

Now registry is fully restored to post-system installation level.

If you want to save your registry data, archive DOS.DAT on the root of your boot drive. In case of corruption, to restore data to this level, do the following:

1. End the RSRV session by:

   a. Editing CONFIG.SYS and remove the following line:

      ```
      RUN=...\RSRV.EXE
      ```

      Note: This line will be added back later.

   b. Restarting the system.

2. Delete all *.DAT files in the root directory of the boot drive:

   ```
   BOOT.DAT DOS.DAT REGISTRY.DAT
   ```

3. Add the following line back in the CONFIG.SYS file:

   ```
   RUN=...\RSRV.EXE
   ```

4. Copy the archived file as DOS.DAT in the root directory.

5. Go to the OS2 directory and type:

   ```
   inst_dos dos
   ```

6. Start up an RSRV session by going to the OS2 directory and typing:

   ```
   start rsrv
   ```

Return Codes

Open32 return codes are currently not guaranteed to be mapped to the same return codes that Windows returns for the same errors. In most cases, this should be acceptable because the set of OS/2 errors is generally a superset of the Windows NT errors. Virtually all of the OS/2 error codes map directly to the identical Windows NT error code. OS/2, however, has several additional error codes that Windows NT does not. These error codes provide more precise information about the cause of the failure.

SMART Conversion Error

When using SMART to convert an RC file that contains a string table with an entry that has an ampersand (&) in it causes the ampersand to be converted incorrectly to a tilde (~) character throughout the entire RC file (including menus and string tables). The string table conversion could cause some unexpected problems with text strings. For example:

```
You & me
```

becomes:

```
You ~ me
```

Structure Packing

The Windows NT DLLs are compiled with 4-byte default packing. However, the Windows NT headers do not force 4-byte default packing. Instead, Windows NT applications are required be compiled with either /Zp4 or no /Zp specification, which defaults to 4-byte packing. This means that if a Win32 application compiles with /Zp1, most of its Win32 structures will be misaligned with respect to the Windows NT DLLs, and the results will be unpredictable.

MSVC has the unique feature of supporting nested packing modes. This means a Win32 application can compile using MSVC with /Zp1 or /Zp2 and still have the Win32 structures properly packed. They simply need to do the following:

```
#pragma pack(4) #include <windows.h> #pragma pack()
```

This will work only for MSVC because only MSVC supports pushing and popping of packing modes. The Win32 headers then, of course, always push or pop the packing mode when they explicitly want to set packing for a particular structure or set of structures. This will not work for any other compiler, including C/Set.

Unlike the Win32 headers, Open32 forces the right packing (4) to occur throughout the Open32 headers, regardless of which compiler options are specified. OS/2 does this to accommodate code bases that are normally compiled using MSVC and to make use of the MSVC packing feature to specify either /Zp1 or /Zp2.

To summarize:

- Open32 applications always get properly packed Win32 structures, regardless of the packing they specify through the compiler switch.
- The true Win32 header files do not do this.
- Open32 must do this to prevent potentially significant changes in Win32 code bases normally compiled with MSVC.

System Handles

The handles for system objects (such as mutexes, events, semaphores, processes, threads, files, and so on) are not compatible with the OS/2 handles.

System Object Names

Windows NT does not allow back slashes within the names of system objects. The OS/2 implementation of system objects does not restrict the name space in any way, so OS/2 does not to support this restriction.

Text Mode Applications

Open32 is designed to work with PM-compatible text mode applications (such as those marked as WINDOWCOMPAT). As with native OS/2 applications, however, a text mode Open32 application is restricted to a limited set of Open32 and PM functions (no GDI and few user functions). Also, as with native OS/2 applications, pure text mode applications (those marked as NOWINDOWCOMPAT) are prohibited from calling any Open32 functions (as well as PM functions). Calls to any Open32 function from an application before PM is initialized are currently defined to produce unpredictable behavior.

Testing

To test Open32 programs, do the following:

1. Rewrite the makefile for the OS/2 build environment.
2. Use the VisualAge C++ compiler to compile the code.
3. Use the IPF compiler included in the Toolkit to compile the help files.
4. Use the RC compiler included in the Toolkit to compile the resource files.
5. Perform testing as you do normally.

Example of Migrating Code

The following is example of the steps you might follow to convert and compile code that uses the sample code-hiworld.c:

Sample

1. Convert the Windows 32-bit resource file hiworld.rc to an OS/2 resource file using SMART.

   Note: SMART will rename the Windows 32-bit resource file from hiworld.rc to hiworld.rcx and creates an OS/2 resource file named hiworld.rc if you click on the Overwrite Original radio button on the Resource Translation dialog. Otherwise, SMART names the OS/2 resource file hiworld.rcx.

2. Change hiworld.h to include hiworld.hhh, which is the output from SMART. hiworld.hhh contains #defines for the resource IDs. For your convenience, this change has been flagged with comments in the Windows 32-bit hiworld.c source code. The comments are removed in the OS/2 hiworld.c code.

3. In any source module that includes <windows.h>, change:

   ```
   #include <windows.h>
   ```

   to:

   ```
   #include <os2win.h>
   ```

   This change has been flagged with comments in the Windows 32-bit hiworld.c code. The comments are removed in the OS/2 hiworld.c code.

4. Change the code in hiworld.c that uses string resource IDs to use MAKEINTRESOURCE with the numeric resource IDs in hiworld.hhh. These have been flagged with comments in the Windows 32-bit hiworld.c source code. The comments are removed in the OS/2 hiworld.c code.

   Note: This is a temporary restriction.

5. Create a DEF file and specify a stack size of at least 128KB. Include a NAME statement which specifies an apptype of WINDOWAPI.

6. Link main.obj (found in the toolkit\c directory under your Open32 directory) with hiworld.obj to form hiworld.exe. This is done in the makefile provided with this sample.

7. Run hiworld.exe as you would any other OS/2 executable file.

Open32 Functions

The highlighted functions behave differently than their Windows 32-bit counterparts.

_lclose
_lcreat
_llseek
_lopen
_lread
_lwrite

AbortDoc
AbortPath
AddAtom
AddFontResource
AdjustWindowRect
AdjustWindowRectEx
AngleArc
AnimatePalette
AppendMenu
Arc
ArcTo
ArrangeIconicWindows

Beep
BeginDeferWindowPos
BeginPaint
BeginPath
BitBlt
BringWindowToTop

CallMsgFilter
CallNextHookEx
CallWindowProc
ChangeClipboardChain
CharLower
CharLowerBuff
CharNext
CharPrev
CharToOem
CharToOemBuff
CharUpper
CharUpperBuff
CheckDlgButton
CheckMenuItem
CheckRadioButton
ChildWindowFromPoint
ChooseColor
ChooseFont
Chord
ClientToScreen
ClipCursor
CloseClipboard
CloseEnhMetaFile
CloseFigure
CloseHandle
CloseMetaFile
CloseWindow
CombineRgn
CommDlgExtendedError
CompareFileTime
ConvertDefaultLocale
CopyAcceleratorTable
CopyCursor
CopyEnhMetaFile
CopyFile
CopyIcon
CopyMetaFile
CopyRect
CountClipboardFormats
CreateAcceleratorTable
CreateBitmap
CreateBitmapIndirect

CreateBrushIndirect
CreateCaret
CreateCompatibleBitmap
CreateCompatibleDC
CreateCursor
CreateDC
CreateDialog
CreateDialogIndirect
CreateDialogIndirectParam
CreateDialogParam
CreateDIBitmap
CreateDIBPatternBrushPt
CreateDirectory
CreateEllipticRgn
CreateEllipticRgnIndirect
CreateEnhMetaFile
CreateEvent
CreateFile
CreateFont
CreateFontIndirect
CreateHatchBrush
CreateIC
CreateIcon
CreateIconFromResource
CreateIconIndirect
CreateMDIWindow
CreateMenu
CreateMetaFile
CreateMutex
CreatePalette
CreatePatternBrush
CreatePen
CreatePenIndirect
CreatePolygonRgn
CreatePolyPolygonRgn
CreatePopupMenu
CreateProcess
CreateRectRgn
CreateRectRgnIndirect
CreateRoundRectRgn
CreateSemaphore
CreateSolidBrush
CreateThread
CreateWindow
CreateWindowEx

DdeAbandonTransaction
DdeAccessData
DdeAddData
DdeClientTransaction
DdeCmpStringHandles
DdeConnect
DdeConnectList
DdeCreateDataHandle
DdeCreateStringHandle
DdeDisconnect
DdeDisconnectList
DdeEnableCallback
DdeFreeDataHandle
DdeFreeStringHandle
DdeGetData
DdeGetLastError
DdeInitialize
DdeKeepStringHandle
DdeNameService
DdePostAdvise
DdeQueryConvInfo
DdeQueryNextServer
DdeQueryString
DdeReconnect
DdeSetUserHandle
DdeUnaccessData
DdeUninitialize

DefDlgProc
DeferWindowPos
DefFrameProc
DefMDIChildProc
DefWindowProc
DeleteAtom
DeleteCriticalSection
DeleteDC
DeleteEnhMetaFile
DeleteFile
DeleteMenu
DeleteMetaFile
DeleteObject
DestroyAcceleratorTable
DestroyCaret
DestroyCursor
DestroyIcon
DestroyMenu
DestroyWindow
DialogBox
DialogBoxIndirect
DialogBoxIndirectParam
DialogBoxParam
DispatchMessage
DlgDirList
DlgDirListComboBox
DlgDirSelectComboBoxEx
DlgDirSelectEx
DllEntryPoint
DosDateTimeToFileTime
DPtoLP
DragAcceptFiles
DragFinish
DragQueryFile
DragQueryPoint
DrawFocusRect
DrawIcon
DrawMenuBar
DrawText
DuplicateHandle

Ellipse
EmptyClipboard
EnableMenuItem
EnableScrollBar
EnableWindow
EndDeferWindowPos
EndDialog
EndDoc
EndPage
EndPaint
EndPath
EnhMetaFileProc
EnterCriticalSection
EnumChildWindows
EnumClipboardFormats
EnumEnhMetaFile
EnumFontFamilies
EnumFonts
EnumMetaFile
EnumObjects
EnumProps
EnumPropsEx
EnumSystemLocales
EnumThreadWindows
EnumWindows
EqualRect
EqualRgn
Escape
ExcludeClipRect
ExcludeUpdateRgn
ExitProcess
ExitThread

ExitWindows
ExitWindowsEx
ExtCreatePen
ExtCreateRegion
ExtFloodFill
ExtSelectClipRgn
ExtTextOut
ExtractIcon

FatalAppExit
FatalExit
FileTimeToDosDateTime
FileTimeToLocalFileTime
FileTimeToSystemTime
FillPath
FillRect
FillRgn
FindAtom
FindClose
FindFirstFile
FindNextFile
FindResource
FindText
FindWindow
FlashWindow
FlattenPath
FloodFill
FlushFileBuffers
FrameRect
FrameRgn
FreeDDElParam
FreeLibrary
FreeProcInstance

GdiComment
GetACP
GetActiveWindow
GetArcDirection
GetAspectRatioFilterEx
GetAsyncKeyState
GetAtomName
GetBitmapBits
GetBitmapDimensionEx
GetBkColor
GetBkMode
GetBoundsRect
GetBrushOrgEx
GetBValue
GetCapture
GetCaretBlinkTime
GetCaretPos
GetCharABCWidths
GetCharWidth
GetClassInfo
GetClassLong
GetClassName
GetClassWord
GetClientRect
GetClipboardData
GetClipboardFormatName
GetClipboardOwner
GetClipboardViewer
GetClipBox
GetClipCursor
GetClipRgn
GetCommandLine
GetCurrentDirectory
GetCurrentObject
GetCurrentPositionEx
GetCurrentProcess
GetCurrentProcessId
GetCurrentThread
GetCurrentThreadId

GetCurrentTime
GetCursor
GetCursorPos
GetDC
GetDCEx
GetDCOrgEx
GetDesktopWindow
GetDeviceCaps
GetDialogBaseUnits
GetDIBits
GetDiskFreeSpace
GetDlgCtrlID
GetDlgItem
GetDlgItemInt
GetDlgItemText
GetDoubleClickTime
GetDriveType
GetEnhMetaFile
GetEnhMetaFileBits
GetEnhMetaFileDescription
GetEnhMetaFileHeader
GetEnhMetaFilePaletteEntries
GetEnvironmentStrings
GetEnvironmentVariable
GetExitCodeProcess
GetExitCodeThread
GetFileAttributes
GetFileInformationByHandle
GetFileSize
GetFileTime
GetFileTitle
GetFileType
GetFocus
GetForegroundWindow
GetFullPathName
GetGraphicsMode
GetGValue
GetIconInfo
GetKerningPairs
GetKeyboardLayout
GetKeyboardState
GetKeyboardType
GetKeyNameText
GetKeyState
GetLastActivePopup
GetLastError
GetLocaleInfo
GetLocalTime
GetLogicalDrives
GetLogicalDriveStrings
GetMapMode
GetMenu
GetMenuCheckMarkDimensions
GetMenuContextHelpID
GetMenuItemCount
GetMenuItemID
GetMenuState
GetMenuString
GetMessage
GetMessageExtraInfo
GetMessagePos
GetMessageTime
GetMetaFile
GetMetaFileBitsEx
GetMiterLimit
GetModuleFileName
GetModuleHandle
GetNearestColor
GetNearestPaletteIndex
GetNextDlgGroupItem
GetNextDlgTabItem
GetNextWindow
GetObject

GetObjectType
GetOEMCP
GetOpenClipboardWindow
GetOpenFileName
GetOutlineTextMetrics
GetOverlappedResult
GetPaletteEntries
GetParent
GetPath
GetPixel
GetPolyFillMode
GetPriorityClass
GetPriorityClipboardFormat
GetPrivateProfileInt
GetPrivateProfileString
GetProcAddress
GetProfileInt
GetProfileString
GetProp
GetQueueStatus
GetRasterizerCaps
GetRegionData
GetRgnBox
GetROP2
GetRValue
GetSaveFileName
GetScrollPos
GetScrollRange
GetStdHandle
GetStockObject
GetStretchBltMode
GetStringTypeEx
GetSubMenu
GetSysColor
GetSystemDirectory
GetSystemMenu
GetSystemMetrics
GetSystemPaletteEntries
GetSystemTime
GetTabbedTextExtent
GetTempFileName
GetTempPath
GetTextAlign
GetTextCharacterExtra
GetTextColor
GetTextExtentPoint
GetTextExtentExPoint
GetTextFace
GetTextMetrics
GetThreadLocale
GetThreadPriority
GetTickCount
GetTimeZoneInformation
GetTopWindow
GetUpdateRect
GetUpdateRgn
GetUserDefaultLangID
GetUserDefaultLCID
GetVersion
GetVersionEx
GetViewportExtEx
GetViewportOrgEx
GetVolumeInformation
GetWindow
GetWindowDC
GetWindowExtEx
GetWindowLong
GetWindowOrgEx
GetWindowPlacement
GetWindowRect
GetWindowsDirectory
GetWindowText
GetWindowTextLength

GetWindowThreadProcessId
GetWindowWord
GetWinMetaFileBits
GetWorldTransform
GlobalAddAtom
GlobalAlloc
GlobalDeleteAtom
GlobalDiscard
GlobalFindAtom
GlobalFlags
GlobalFree
GlobalGetAtomName
GlobalHandle
GlobalLock
GlobalMemoryStatus
GlobalReAlloc
GlobalSize
GlobalUnlock

HeapAlloc
HeapCreate
HeapDestroy
HeapFree
HeapReAlloc
HeapSize
HideCaret
HiliteMenuItem

InflateRect
InitAtomTable
InitializeCriticalSection
InSendMessage
InsertMenu
InterlockedDecrement
InterlockedExchange
InterlockedIncrement
IntersectClipRect
IntersectRect
InvalidateRect
InvalidateRgn
InvertRect
InvertRgn
IsBadCodePtr
IsBadHugeReadPtr
IsBadHugeWritePtr
IsBadReadPtr
IsBadStringPtr
IsBadWritePtr
IsCharAlpha
IsCharAlphaNumeric
IsCharLower
IsCharUpper
IsChild
IsClipboardFormatAvailable
IsDBCSLeadByte
IsDialogMessage
IsDlgButtonChecked
IsIconic
IsMenu
IsRectEmpty
IsValidLocale
IsWindow
IsWindowEnabled
IsWindowVisible
IsZoomed

KillTimer

LeaveCriticalSection
LineDDA
LineTo
LoadAccelerators
LoadBitmap

LoadCursor
LoadIcon
LoadLibrary
LoadMenu
LoadMenuIndirect
LoadModule
LoadResource
LoadString
LocalAlloc
LocalDiscard
LocalFileTimeToFileTime
LocalFlags
LocalFree
LocalHandle
LocalLock
LocalReAlloc
LocalSize
LocalUnlock
LockFile
LockResource
LockWindowUpdate
LPtoDP
lstrcat
lstrcmpW
lstrcpy
lstrlen

MakeProcInstance
MapDialogRect
MapWindowPoints
MaskBlt
MessageBeep
MessageBox
ModifyMenu
ModifyWorldTransform
MoveFile
MoveToEx
MoveWindow
MsgWaitForMultipleObjects
MulDiv
MultiByteToWideChar

OemToChar
OemToCharBuff
OffsetClipRgn
OffsetRect
OffsetRgn
OffsetViewportOrgEx
OffsetWindowOrgEx
OpenClipboard
OpenEvent
OpenFile
OpenMutex
OpenProcess
OpenSemaphore
OutputDebugString

PackDDElParam
PaintRgn
PatBlt
PathToRegion
PeekMessage
Pie
PlayEnhMetaFile
PlayEnhMetaFileRecord
PlayMetaFile
PlayMetaFileRecord
PolyBezier
PolyBezierTo
PolyDraw
Polygon
Polyline
PolylineTo

PolyPolygon
PolyPolyline
PostMessage
PostQuitMessage
PostThreadMessage
PrintDlg
PtInRect
PtInRegion
PtVisible
PulseEvent

ReadFile
RealizePalette
Rectangle
RectInRegion
RectVisible
RedrawWindow
Registry functions (see Registry and Registry Functions):

- RegCloseKey
- RegCreateKey
- RegCreateKeyEx
- RegDeleteKey
- RegDeleteValue
- RegEnumKey
- RegEnumKeyEx
- RegEnumValue
- RegisterClass
- RegisterClipboardFormat
- RegisterWindowMessage
- RegOpenKey
- RegOpenKeyEx
- RegQueryInfoKey
- RegQueryValue
- RegQueryValueEx
- RegSetValue
- RegSetValueEx

ReleaseCapture
ReleaseDC
ReleaseMutex
ReleaseSemaphore
RemoveDirectory
RemoveFontResource
RemoveMenu
RemoveProp
ReplaceText
ReplyMessage
ResetDC
ResetEvent
ResizePalette
RestoreDC
ResumeThread
ReuseDDElParam
RoundRect

SaveDC
ScaleViewportExtEx
ScaleWindowExtEx
ScreenToClient
ScrollDC
ScrollWindow
ScrollWindowEx
SearchPath
SelectClipRgn
SelectObject
SelectPalette
SendDlgItemMessage
SendMessage
SetAbortProc
SetActiveWindow
SetArcDirection
SetBitmapBits
SetBitmapDimensionEx
SetBkColor

SetBkMode
SetBoundsRect
SetBrushOrgEx
SetCapture
SetCaretBlinkTime
SetCaretPos
SetClassLong
SetClassWord
SetClipboardData
SetClipboardViewer
SetCurrentDirectory
SetCursor
SetCursorPos
SetDIBits
SetDIBitsToDevice
SetDlgItemInt
SetDlgItemText
SetDoubleClickTime
SetEndOfFile
SetEnhMetaFileBits
SetEnvironmentVariable
SetEvent
SetFileAttributes
SetFilePointer
SetFileTime
SetFocus
SetForegroundWindow
SetGraphicsMode
SetHandleCount
SetKeyboardState
SetLastError
SetLocaleInfo
SetLocalTime
SetMapMode
SetMapperFlags
SetMenu
SetMenuContextHelpID
SetMenuItemBitmaps
SetMetaFileBitsEx
SetMiterLimit
SetPaletteEntries
SetParent
SetPixel
SetPolyFillMode
SetPriorityClass
SetProp
SetRect
SetRectEmpty
SetRectRgn
SetROP2
SetScrollPos
SetScrollRange
SetStdHandle
SetStretchBltMode
SetSysColors
SetSystemTime
SetTextAlign
SetTextCharacterExtra
SetTextColor
SetTextJustification
SetThreadPriority
SetTimer
SetTimeZoneInformation
SetViewportExtEx
SetViewportOrgEx
SetVolumeLabel
SetWindowExtEx
SetWindowLong
SetWindowOrgEx
SetWindowPlacement
SetWindowPos
SetWindowsHookEx
SetWindowText

SetWindowWord
SetWinMetaFileBits
SetWorldTransform
ShellExecute
ShowCaret
ShowCursor
ShowOwnedPopups
ShowScrollBar
ShowWindow
SizeofResource
Sleep
StartDoc
StartPage
StretchBlt
StretchDIBits
StrokeAndFillPath
StrokePath
SubtractRect
SuspendThread
SwapMouseButton
SystemParametersInfo
SystemTimeToFileTime
SystemTimeToTzSpecificLocalTime

TabbedTextOut
TerminateProcess
TerminateThread
TextOut
timeGetSystemTime
timeGetTime
TlsAlloc
TlsFree
TlsGetValue
TlsSetValue
TrackPopUpMenu
TranslateAccelerator
TranslateMDISysAccel
TranslateMessage

UnhookWindowsHookEx
UnionRect
UnlockFile
UnpackDDElParam
UnrealizeObject
UnregisterClass
UpdateWindow

ValidateRect
ValidateRgn
VkKeyScan

WaitForInputIdle
WaitForMultipleObjects
WaitForSingleObject
WaitMessage
WideCharToMultiByte
WidenPath
WindowFromDC
WindowFromPoint
WinExec
WinHelp
WinMain
WriteFile
WritePrivateProfileString
WriteProfileString
wsprintf
wvsprintf

Yield

ZeroMemory

Open32-Unique Functions

The functions HPSToHDC and DeleteHDCObjects are unique to Open32 and do not have a Windows counterpart function. Descriptions of each of these functions follow.

HPSToHDC

This function will set up the HPS for Open32 application use and return a Windows HDC handle. The Open32 application can then use this handle to perform drawing operations. The applications should call DeleteHDCObjects when finished using the handle.

Syntax:

```
     HPSToHDC (HWND, HPS, PSZ, LPRECT)
```

Parameters:

HWND	The handle of the window associated with the HPS.
HPS	The HPS to be converted to a Windows HDC.
PSZ	The string containing the device name of the printer associated with the HPS or NULL for a non-printer HPS.
LPRECT	Rectangle that contains the coordinates for the drawing area.

DeleteHDCObjects

This function deletes Open32 objects that were created when this HDC was converted for Open32 use.

Syntax:

```
void DeleteHDCObjects (HDC)
```

Parameters:

HDC	The HDC to be deleted.

Notices

November 1997

The following paragraph does not apply to the United Kingdom or any country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This publication could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time.

It is possible that this publication may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

Requests for technical information about IBM products should be made to your IBM reseller or IBM marketing representative.

Copyright Notices

Disclaimers

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY 10594
U.S.A.

Asia-Pacific users can inquire, in writing, to the IBM Director of Intellectual Property and Licensing, IBM World Trade Asia Corporation, 2-31 Roppongi 3-chome, Minato-ku, Tokyo 106, Japan.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Department LZKS, 11400 Burnet Road, Austin, TX 78758 U.S.A. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

DB2
IBM
OS/2
Presentation Manager (PM)
VisualAge

The following terms are trademarks of other companies:

OpenDoc	Apple Computer, Inc.
SMART	One Up Corporation
